MODELING MISSING COVARIATE DATA AND TEMPORAL FEATURES OF TIME-DEPENDENT COVARIATES IN TREE-STRUCTURED SURVIVAL ANALYSIS

by

Meredith JoAnne Lotz

B.A., St. Olaf College, 2004

Submitted to the Graduate Faculty of

the Graduate School of Public Health in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2009

UNIVERSITY OF PITTSBURGH

GRADUATE SCHOOL OF PUBLIC HEALTH

DEPARTMENT OF BIOSTATISTICS

This dissertation was presented

by

Meredith JoAnne Lotz

It was defended on

June 12, 2009

and approved by

Stewart Anderson, Ph.D., Professor, Department of Biostatistics,

Graduate School of Public Health, University of Pittsburgh

Lan Kong, Ph.D., Assistant Professor, Department of Biostatistics,

Graduate School of Public Health, University of Pittsburgh

Sati Mazumdar, Ph.D., Professor, Department of Biostatistics,

Graduate School of Public Health, University of Pittsburgh

Benoit Mulsant, M.D., Professor, Department Psychiatry,

School of Medicine, University of Pittsburgh

Dissertation Director: Stewart Anderson, Ph.D., Professor, Department of Biostatistics,

Graduate School of Public Health, University of Pittsburgh

MODELING MISSING COVARIATE DATA AND TEMPORAL FEATURES OF TIME-DEPENDENT COVARIATES IN TREE-STRUCTURED SURVIVAL ANALYSIS

Meredith JoAnne Lotz, PhD

University of Pittsburgh, 2009

Tree-structured survival analysis (TSSA) is used to recursively detect covariate values that best divide the sample into subsequent subsets with respect to a time to event outcome. The result is a set of empirical classification groups, each of which identifies individuals with more homogeneous risk than the original sample. We propose methods for managing missing covariate data and also for incorporating temporal features of repeatedly measured covariates into TSSA. First, for missing covariate data, we propose an algorithm that uses a stochastic process to add draws to an existing single tree-structured imputation method. Secondly, to incorporate temporal features of repeatedly measured covariates, we propose two different methods: (1) use a two-stage random effects polynomial model to estimate temporal features of repeatedly measured covariates to be used as TSSA predictor variables, and (2) incorporate other types of functions of repeatedly measured covariates into existing time-dependent TSSA methodology. We conduct simulation studies to assess the accuracy and predictive abilities of our proposed methodology. Our methodology has particular public health importance because we create, interpret and assess TSSA algorithms that can be used in a clinical setting to predict response to treatment for late-life depression.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

Many thanks my advisor, Dr. Anderson, as well as my committee members, Drs. Mazumdar, Kong, and Mulsant. I greatly appreciate all of your help, encouragement, and guidance throughout the last five years.

1.0 INTRODUCTION

Late-life depression can lead to serious health consequences, including emotional suffering, caregiver strain, disability associated with medical and cognitive disorders and increased mortality due to suicide [1]. Although some older adults are successfully treated for depression through acute and maintenance therapy, others do not respond to treatments for depression, at which point a different regime may be employed. However, if a clinician is able to identify patient risk early in the clinical course, their treatments may be tailored accordingly, possibly resulting in a better clinical outcome.

Tree-structured survival analysis (TSSA) is one method that can help clinicians identify a patient's risk and subsequently personalize their treatment. A fundamental goal of TSSA is to create empirical subgroups that distinguish time to clinical events based on their covariate values. The resulting model is clinically meaningful because it can be visually represented by a single tree-structured algorithm. Using this algorithm, the clinician can ask a series of simple yes-or-no questions regarding clinical characteristics and subsequently classify a patient into one of the identified risk groups. Each of these risk groups may be associated with a treatment strategy, resulting in an efficient tool to assist the clinician in making a treatment decision.

In this dissertation, we present methodologies for TSSA that were originally motivated by a clinical trial for maintenance therapies in late-life depression (MTLD) by Reynolds et al. [1]. In this clinical trial, described thoroughly in Chapter 4, individuals were first treated for depression in an acute phase, and later randomized to maintenance therapies in a 2×2 block design with a placebo drug versus nortriptyline and a placebo medical clinic versus interpersonal therapy. We desire to use the acute phase of this study, which includes weekly follow-up for each individual, to create an algorithm that can be utilized

by clinicians to classify patients into risk groups based on their updated covariate values and subsequently personalize their treatment. However, there are two characteristics of the data that require novel methodology in order to create accurate and useful tree models. First, there are many missing covariate observations, both at baseline and during follow-up, which cannot be modeled appropriately through currently used methods. Second, we hypothesize that temporal features of each individual's clinical characteristics could provide valuable prognostic information, and should therefore be utilized within our tree-structured algorithm. Hence, our proposed dissertation methodology, which is motivated by these challenges, is focused in two areas of TSSA: (1) missing covariate data, and (2) including temporal features of time-dependent covariates.

1.1 MODELING MISSING COVARIATE DATA IN TSSA

Tree-structured methodology allows one to classify outcomes with respect to covariates which themselves could have extreme outliers, skewed distributions and/or nonlinear relationships with the outcome. Therefore, any method selected to model missing observations in such covariates should also be able to accommodate these characteristics. In addition to this requirement, we believe that any method utilized for missing covariate data in TSSA should also output one clinically meaningful algorithm, as opposed to a "black box" that does not allow the user to visualize the classification process. Current methods used for modeling missing covariate data in binary tree-structured algorithms, such as utilizing multiple imputation with bagging techniques [2], single regression imputation methods [3], or surrogate splitting [4] can each only accommodate a subset of these desired features.

In Chapter 5, we propose a method for modeling missing covariate data that accommodates the desired features outlined above and thus is ideal for use with tree-structured survival analysis. Our method is a multiple imputation algorithm that adds draws of stochastic error to a tree-based single imputation method previously presented by Conversano and Siciliano [5]. We perform a simulation study to assess the accuracy of our stochastic multiple imputation method when covariate data are missing at random and compare our algorithm

to other currently used methods for accommodating missing covariate data. We also utilize our methodology create a tree-structured survival model based on the MTLD dataset.

1.2 INCLUDING TEMPORAL FEATURES OF TIME-DEPENDENT COVARIATES IN TSSA

Traditionally, only fixed time covariates are used to create tree models. However, covariate values often change over time, and utilizing these updated values may help to improve the accuracy and reliability of the prognostic groups selected by the model. Both Fisher and Lin [6] and Gail [7] discuss that an integral component of using repeatedly measured covariates in the Cox model is selecting the type of time-dependence, and we believe that this may also true with respect to the creation of classification groups in TSSA. Specifically, although some repeatedly measured covariates can be used as observed to classify an individual into a risk group, others are more meaningful when various temporal features of their observed repeatedly measured values are incorporated. We propose two different methods for including temporal features of repeatedly measured covariates in TSSA.

In our first method, presented in Chapter 6, we propose using the two-stage random-effects model can be used to extract estimates of individual temporal features of repeatedly measured covariates, e.g., rate of change or curvature over time. The set of these individual temporal features is then used to predict the survival outcome through a TSSA model. The result is an algorithm that designates prognostic groups based on not only fixed time covariate values but also a selected temporal feature of a repeatedly measured covariate. For illustration, we use our model to predict the time to treatment response in older adults based on their linear rate of change of anxiety during the acute phase of the MTLD study. The empirical Brier score [8] is used to compare the predictive abilities of our proposed model to a traditional fixed-time covariate model when used to predict treatment response throughout the course of treatment.

In our second method, presented in Chapter 7, we propose to create time-dependent functionals of time-dependent clinical characteristics and use them as covariates in a time-

dependent TSSA model. Both Bacchetti and Segal [9] and Huang, Chen et al. [10] developed time-dependent TSSA methodology using the two-sample rank statistic and the piecewise exponential distribution, respectively. However, in order to utilize time-dependent functionals, there are a number of additional issues that need to be addressed, largely due to the fact that these functionals may be non-monotonically increasing or decreasing with time. Although the current time-dependent TSSA methodology by Bacchetti and Segal [9] and Huang, Chen et al. [10] can technically handle non-monotonically increasing or decreasing covariates, the vast majority of their work has focused on testing and applying the much simpler scenario where time-dependent covariates are increasing or decreasing monotonically. Therefore, we first address the general problem of incorporating non-monotonically increasing or decreasing time-dependent covariates through a simulation study. Next, for illustration of our proposed methodology, we use time-dependent TSSA to predict treatment response in older adults based on the non-monotonically changing functionals of time-dependent anxiety. The empirical Brier score [8] is used to compare the predictive abilities of our various models when used to classify patients into risk groups throughout the course of acute treatment.

2.0 LITERATURE REVIEW: SURVIVAL ANALYSIS

2.1 NOTATION

Each individual, i, $i = 1, \ldots, N$, is associated with a true event time, T_i, and a true time of censoring from the study, C_i. The observed outcome for each individual is denoted by (T_i^*, δ_i), where $T_i^* = \min(T_i, C_i)$, $\delta_i = 1$ when $T_i \leq C_i$ and $\delta_i = 0$ otherwise. At each possible discrete observed event time, t_j, $j = 1, \ldots, J$, we observe the set of R_j individuals at risk just prior to event time t_j, denoted by $\mathcal{R}_j$, and also the set of D_j individuals who have an event at time t_j, denoted by $\mathcal{D}_j$.

2.2 SURVIVAL QUANTITIES

The fundamental goal of survival analysis is to model the time until an event, represented by the random variable T. The event associated with T can take on many different forms, such as time until death, time until recovery, or time until the presence of disease is detected. Regardless of the type of event, T can be modeled with the survival function

$$S(t) = Pr(T > t), \tag{2.1}$$

which represents the probability that the event, T, has not yet occurred by time t. This survival distribution is closely related to the cumulative distribution function (c.d.f.) and probability distribution function (p.d.f.) of T, both of which can be shown through the relationship

$$S(t) = 1 - F(t) = \int_t^\infty f(x)\, dx, \tag{2.2}$$

where $F(t)$ represents the c.d.f. and $f(t)$ represents the p.d.f. of T [11].

To make inferences regarding the behavior of T, it is first necessary to estimate the survival function using observed data from a sample. One such estimator is the Kaplan-Meier product limit estimator [12],

$$\hat{S}(t) = \begin{cases} 1 & \text{if } t < t_1 \\ \prod_{t_j \leq t}[1 - \frac{D_j}{R_j}] & \text{if } t_1 \leq t \end{cases}, \tag{2.3}$$

which is defined for all values of t for which there are outcome data.

An alternative characterization of the distribution of T is the hazard function, which is related to the instantaneous rate of failure. The hazard function is defined as

$$\lambda(t) \simeq \lim_{\Delta t \to \infty} \frac{P[T \leq t + \Delta t | T \geq t]}{\Delta t}, \tag{2.4}$$

where $\lambda(t)\Delta t$ is the approximate probability of failure in the next small interval, from t to $t + \Delta t$, given survival until time t. The hazard function (2.4), p.d.f., and survival function (2.1) are related by the equation [11]

$$\lambda(t) = \frac{f(t)}{S(t)}. \tag{2.5}$$

The cumulative hazard function can be derived from the survival distribution using the formula

$$\Lambda(t) = -ln[S(t)]. \tag{2.6}$$

Thus, an estimate for the cumulative hazard at a time t could be derived from the Kaplan Meier product limit estimator (2.3). However, the cumulative hazard function can also be modeled with the Nelson-Aalen estimate, which has better small-sample-size performance. This estimate is expressed as

$$\hat{\Lambda}(t) = \begin{cases} 1 & \text{if } t < t_1 \\ \sum_{t_j \leq t}[\frac{D_j}{R_j}] & \text{if } t_1 \leq t \end{cases}. \tag{2.7}$$

Just like the product-limit estimate of the survival function (2.3), the Nelson-Aalen estimate of the cumulative hazard function (2.7) is also defined up to the largest observed time in the study. Both the Kaplan Meier product limit estimate and the Nelson Aalen estimate are based on the assumption of non-informative censoring. This assumption implies that knowledge of censoring time for an individual provides no further information about the person's likelihood of survival at a future time had the individual continued in the study [11].

2.3 THE COX PROPORTIONAL HAZARDS MODEL

2.3.1 Fixed-Time Covariates

Although it can be very informative to simply estimate the hazard or survival of one sample, it is often more clinically relevant to model it using explanatory covariates. One of the most widely used models that delineates the relationship between a set of covariates and a time to event outcome is the Cox proportional hazards model [13, 14]. Formally, it models the hazard rate of a time-to-event outcome at time t_j, conditioned on a set covariates, as

$$\lambda(t_j|\mathbf{X}) = \lambda_0(t_j)e^{(\mathbf{X}\boldsymbol{\beta})}, \tag{2.8}$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]'$ represents the vector of regression coefficients and $\mathbf{X} = \{x_{ik}\}$, where $i = 1, \ldots, n$ individuals and $k = 1, \ldots p$ covariates. In this model, $\lambda_0(t_j)$ is an unspecified function defining the hazard rate when $\mathbf{X} = \mathbf{0}$, and is usually referred to as the baseline hazard rate.

One advantage of the Cox model (2.8) is that it makes no assumptions about the probability distribution of the hazard, λ_0. However, since $\mathbf{X}$ is not a function of time, the underlying assumption is that the hazards in different covariate groups are proportional over time and also that all individuals share the same baseline hazard rate, $\lambda_0(t_j)$.

The Cox partial likelihood, denoted by $L(\boldsymbol{\beta})$, is constructed to estimate the parameters $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]'$ from the Cox proportional hazards model (2.8). To account for discrete time points and include ties we utilize the sum of the covariates of the individuals with events at time t_j, denoted $\mathbf{s}_j = \sum_{i \in \mathcal{D}_j} X_i$, where $X_i = [x_{i1}, \ldots, x_{ip}]$ [11]. This likelihood is expressed as

$$L(\boldsymbol{\beta}) \;=\; \prod_{j=1}^{J} \frac{\exp\left(\mathbf{s}_j\boldsymbol{\beta}\right)}{\left[\sum_{i \in \mathcal{R}_j} \exp\left(X_i\boldsymbol{\beta}\right)\right]^{\delta_j}}. \tag{2.9}$$

Note that the baseline hazard, $\lambda_0(t_j)$, remains unspecified in this likelihood. As a result, the Cox partial likelihood is only explicitly a function of the covariates and the regression coefficients. The numerator depends only on information from the individuals who experienced

the event at time t_j, whereas the denominator utilizes information from the all individuals who have not yet experienced the event just prior to time t_j. Maximum likelihood estimates for the p regression parameters, $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]'$, can be obtained through maximization of the Cox partial likelihood with respect to the parameters of interest.

After obtaining the maximum likelihood estimates from the Cox partial likelihood (2.9), it may be of interest to be use these model estimates to predict the survival probability of an individual, i, who may or may not have been in the original cohort. This prediction is accomplished via the prognostic index,

$$PI_i = x_{i1}\hat{\beta}_1 + \ldots + x_{ip}\hat{\beta}_p = X_i\hat{\boldsymbol{\beta}}, \tag{2.10}$$

which is defined as the sum of the covariate values of individual i, weighted by the corresponding estimated regression coefficients.

2.3.2 Time-Dependent Covariates

The Cox proportional hazards model can be extended to include repeatedly measured covariates by modifying the Cox model for fixed covariates (2.8) as

$$\lambda(t_j|\mathbf{W}(t_j)) = \lambda_0(t_j)\exp\Big(\mathbf{W}(t_j)\boldsymbol{\beta}\Big), \tag{2.11}$$

where $\mathbf{W}(t_j) = \{w_{ik}(t_j)\}$, with $i = 1, \ldots, n$ individuals and $k = 1, \ldots, p$ covariates. The maximum likelihood estimates for this model are obtained by rewriting the partial likelihood in equation 2.9 as

$$L(\boldsymbol{\beta}) \;=\; \prod_{j=1}^{J} \frac{\exp\Big(\mathbf{s}(t_j)\boldsymbol{\beta}\Big)}{\left[\sum_{i\in\mathcal{R}(t_j)}\exp\Big(W_i(t_j)\boldsymbol{\beta}\Big)\right]}, \tag{2.12}$$

where $s(t_j) = \sum_{i\in\mathcal{D}_j} W_i(t_j)$ and $W_i(t_j) = [w_{i1}(t_j), \ldots, w_{ip}(t_j)]$ is the vector of $k = 1, \ldots, p$ covariate observations for an individual, i, at time t_j. In this partial likelihood, the covariate values in both the numerator and the denominator may differ for each discrete event time, t_j, accommodating covariate changes that occur over time. Due to the more complicated likelihood, obtaining maximum likelihood estimates for regression parameters is much more

computationally intensive and is usually accomplished using a Newton Raphson algorithm or another iterative technique [11].

Unfortunately, prediction via the prognostic index (2.10) is not as straightforward when repeatedly measured covariates are used. In these situations, the ability to predict is often lost because the model depends on a changing covariate value, and future values of this covariate may be unknown. When we do know a future covariate value, we also know that the individual has not yet reached the endpoint at this future time. Thus, knowing the covariate implies knowledge of the failure status of the patient. Due to these complications, predictions using repeatedly measured covariates from the Cox model should be made with caution [6].

2.3.3 Classification Using the Cox Model

The Cox proportional hazards model has many benefits and can be extremely useful for modeling time to event data. Unfortunately, when the aim of the analysis is to create prognostic classification groups, these analyses may not be the best choice for a number of reasons. First, the Cox model requires the computation of each person's individual risk, which can be cumbersome to apply for every new case. Second, although a given Cox model may help the clinician identify the risk associated with a set of covariates, the interpretation of these covariates in relation to the disease process may be difficult. Third, extensive and time consuming analyses may need to be performed to identify important interactions between the covariates. And fourth, if prognostic classification groups are to be created using the Cox model, ad hoc cutoff points for covariates must be chosen. This is because the Cox model provides no information regarding which split points best differentiate patient risk. Due to these limitations, tree-based survival analysis may be more appropriate when the research goal is to create an algorithm to classify new individuals into prognostic groups [8].

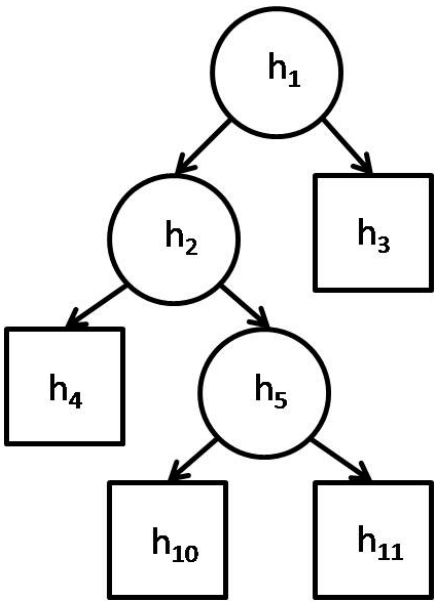

Figure 2.1: Example of binary tree

2.4 TREE-BASED SURVIVAL ANALYSIS

2.4.1 Overview

Growing a binary tree, such as the example in Figure 2.1, consists of recursively splitting the predictor space into disjoint binary subsets, or nodes, h. The splitting begins with the entire predictor space, $\mathcal{X}$, also called the root node, h_1. A rule is used to split the root node into two disjoint daughter nodes, which may each be split into two more daughter nodes, and so on. After the nodes each have a minimum number of observations, or are all of the same class, the splitting ceases and the maximum tree is created. At this point, it is often useful to prune the tree to create a more parsimonious model. Of particular interest in the final model, H is the set of terminal nodes, $\tilde{H}$, which represent empirically selected classification groups.

2.4.2 History

Although tree-based methods are now widely used in survival analysis, they were originally conceptualized and implemented by Morgan and Sonquist [15] to facilitate the detection of interactions in an algorithm called AID (Automatic Interaction Detection). It was not until almost two decades later that Ciampi et al. [16] and Marubini et al. [17] adopted the idea of tree based modeling and applied it to survival data. They created subgroups based on the distance between survival curves and the Cox model, respectively.

Breiman, et al. [18] were the first to develop a widely used tree-based algorithm, presented as a comprehensive book and user-friendly computer software. Their algorithm, "Classification and Regression Trees" (CART), uses measures of within-node variance to recursively split a sample into more homogeneous subsets with respect to a continuous, nominal, or ordinal outcome variable. CART brought many improvements to tree-based modeling. One of the most important of these improvements is their efficient and automated cost-complexity pruning algorithm, which allows the user to select a more parsimonious subtree from the original full-sized tree.

After the release of CART, methods for tree-based regression and classification were more accessible and better documented, and subsequently much more widely used. Unfortunately, methods for TSSA were not developed in the original CART methodology and thus needed to be more comprehensively documented and evaluated before they could also be widely used. Gordon and Olshen [19] were the first to combine tree-based survival analysis with CART's within-node splitting statistic and pruning algorithm. They proposed a Kaplan-Meier estimate of the survival function to select the split that maximizes the distance between estimated survival functions. Davis and Anderson [20] assumed an exponential survival model characterized by the constant hazard function, and used exponential log-likelihood loss to choose the best split. LeBlanc and Crowley [21] also used the likelihood, employing one-step full exponential likelihood deviance to estimate the parameters of interest. Ahn and Loh [22] took a different approach and developed a method for fitting piecewise exponential proportional hazards models to censored survival data. Their within-node goodness of split test is based on studying the patterns of the Cox residuals along each covariate axis.

For survival outcomes, Segal [23] was the first to develop a method that utilizes measures of between-node separation instead of within-node homogeneity. He proposed that any two-sample rank statistic of either the Tarone-Ware [24] or Harrington-Fleming [25] family could be chosen as a between-node splitting statistic. Although one appeal of this approach was that it could easily handle censored survival data, a complication was that CART's pruning algorithm could not be directly applied because a within-node homogeneity criterion was not used to select the best split. To circumvent this problem Segal proposed a non-automated pruning algorithm. Later, LeBlanc and Crowley [26] developed automated pruning algorithms adapted for trees that split on between-node criteria. One such algorithm uses a training/test dataset approach. Another algorithm, for use with smaller sample sizes, employs a bootstrapping method to prune trees.

Although the CART algorithm does not allow users to model time-to-event outcomes, there does exist software that accommodate such data outcomes. Ciampi et al. [27] created RECPAM for censored survival data; its uniqueness came in the form of recombining terminal nodes that were too similar. Later adaptations to their program allowed both likelihood based splitting (measuring within-node homogeneity) and two-sample nonparametric splitting (measuring between node differences) [28]. However, a disadvantage of RECPAM is that not all of the pruning algorithms are automated. Therneau, Atkinson, et al. developed RPART [29], an algorithm in R [30] which could create tree-based regression, classification, and survival analyses models [4]. Their survival analysis splitting algorithm uses LeBlanc and Crowley's full exponential likelihood deviance criterion [21]. One benefit of RPART is that cross-validation methods to prune tree are all automated, making it a very efficient program. Both the RPART and RECPAM programs are widely used and extremely beneficial for TSSA. Unfortunately, neither program allows for the user to utilize a between-node splitting criterion, as proposed by Segal [23] and LeBlanc and Crowley [26], as well as prune the resulting tree.

2.4.3 Growing the Tree

A set of rules is defined to create empirically selected splits. Possible splits on the predictor space, $\mathcal{X}_h$, at a node, h, are induced by any question of the form "Is $X \in \mathcal{S}$?", where $\mathcal{S} \subset \mathcal{X}_h$. If X is an ordered variable, the split may be created by a question of the form: "Is $X \leq c$?". If X is, nominal with values in $\mathcal{B} = \{b_1, b_2, \ldots, b_r\}$, the split may be created by a question of the form "Is $X \in \mathcal{S}$?" where $\mathcal{S} \subset \mathcal{B}$ [18, 26]. At each node h, there are many possible splits, s_h, that could divide the predictor space, $\mathcal{X}_h$, into two disjoint subsets. The set of all possible splits at node h is denoted by $\mathcal{S}_h$.

To judge the quality of each potential split, $s_h \in \mathcal{S}_h$, a splitting criterion, $G(s_h)$, is used to evaluate the predictive improvement that could result from that particular division of the predictor space, $\mathcal{X}_h$. The splitting criterion, $G(s_h)$, is computed for each possible split point, $s_h \in \mathcal{S}_h$, and the best split for the node, h, is the split (s_h^*) such that

$$G(s_h^*) = \max_{s_h \in \mathcal{S}_h} G(s_h). \tag{2.13}$$

This best split, s_h^*, is the split selected to divide the node, h, into the left and right daughter nodes (h_L and h_R, respectively). Each daughter node may subsequently go through the same splitting process based on its respective predictor space. Typically, the splitting on each node continues until the largest possible tree, H_{MAX}, has been created. H_{MAX} occurs when the nodes to be split are of a prespecified minimum size or if all the observations in each node are of the same class and cannot be differentiated further. The nodes that are not split any further are called terminal nodes; the set of these terminal nodes in a tree, H, is denoted $\tilde{H}$ [18, 26].

The selection of a splitting criterion, $G(s_h)$, depends largely on the nature of the outcome of interest, and for our purposes we are most interested in survival outcomes. However, the splitting algorithm we utilize was first developed for categorical and continuous outcomes by Breiman et al. [18]. For a continuous outcome, y, Breiman et al. developed regression trees based on a within-node splitting criterion that measures the decrease in squared deviance resulting from creating two nodes, h_L and h_R, based on a split, s_h. This statistic is calculated as

$$G(s_h) = G(h) - \big(G(h_L) + G(h_R)\big), \tag{2.14}$$

where

$$G(h) = \frac{1}{N_h} \sum_{i \in \mathcal{L}_h} \left(y_i - \frac{1}{N_h} \sum_{i \in \mathcal{L}_h} y_i \right)^2. \tag{2.15}$$

Here, y_i is the continuous outcome for the i^{th} individual and $\mathcal{L}_h$ is the set of N_h individuals at node h. The best split, s_h^*, results in the greatest decrease in squared deviance, as defined by equation (2.13).

When censored survival data are used in a tree-structured analysis, a split statistic that will account for the censoring should be chosen. Numerous splitting statistics for censored survival data can be used. Parametric statistics based on the likelihood, semi-parametric statistics based on the Cox model, and nonparametric statistics based on a two-sample rank test are all viable options and may be appropriate in certain situations. In the following subsections we discuss two such statistics in detail, the family of two-sample rank statistic and the full exponential likelihood deviance statistic.

2.4.3.1 Two-Sample Rank Statistics Segal first proposed using a two-sample rank statistic as a splitting criterion for TSSA, stating many benefits for this choice. Specifically, the best predictor and cutoff value are invariant to monotone transformations, the models created are less sensitive to outliers due to its nonparametric nature, it is associated with simple and efficient splitting algorithms, and it can easily incorporate censored survival data [23].

Segal [23] and LeBlanc and Crowley [26] suggest that any statistic from either the Tarone-Ware [24] or Harrington-Fleming family [25] can be used as a non-parametric two-sample rank test for censored data. Statistics from these families judge the quality of a possible split, s_h, by quantifying the difference between the two survival distributions created by the split.

The statistic for the Tarone-Ware class of two-sample rank statistics, as used by Segal [9], is obtained by constructing a sequence of 2×2 tables, each one representing a distinct event

time, t_j. For the j^{th} distinct event time, the following table is constructed

	Event	No Event	
Left Node	x_j		m_{j1}
Right Node			
	n_{j1}		n_j

The splitting statistic based on the Tarone-Ware class of two-sample statistics is calculated using information from each table constructed at discrete event time points t_j, $j = 1, \ldots, J$, as follows:

$$G(s_h) = \frac{\sum_{j=1}^{J} w_j[x_j - E_o(X_j)]}{\sqrt{[\sum_{j=1}^{J} w_j^2 Var_o(X_j)]}}, \tag{2.16}$$

where w_j is the weight for time t_j. Under the null hypothesis that the failure rates of the left and right daughter nodes are equal, the random variable for the number of events occurring in the left node at time t_j, X_j, follows the hypergeometric distribution, such that

$$E_0(X_j) = \frac{m_{j1}n_{j1}}{n_j} \tag{2.17}$$

and

$$Var_0(X_j) = \frac{m_{j1}n_{j1}(n_j - m_{j1})(n_j - n_{j1})}{(n_j - 1)n_j^2}. \tag{2.18}$$

The split selected to divide the data is the split that results in the largest between-node separation, as defined by equation 2.13.

Altering the weight, w_j, in the Tarone-Ware rank statistic may result in different two-sample rank statistics. For example, setting $w_j = 1$ at time t_j, for all $j = 1, \ldots, J$, results in the log rank test, proposed by Mantel [31] and Cox [13]. Setting $w_j = n_j$ at time t_j, for all $j = 1, \ldots, J$, results in the generalized Wilcoxon test of Gehan [32] and Breslow [33]. The Tarone Ware class of statistics follows an asymptotic chi-square with $r - 1$ degrees of freedom, where r is equal to the number of groups being compared. For our purposes, we have $r = 2$ groups being compared (the left and right daughter nodes) and thus the Tarone Ware statistic is distributed as χ_1^2 [24].

2.4.3.2 One-Step Full Exponential Likelihood Deviance

LeBlanc and Crowley [21] propose a one-step full exponential likelihood deviance criterion to select the best split at a node, h. Their method adopts a proportional hazard model,

$$\lambda_h(t) = \theta_h \lambda_0(t), \tag{2.19}$$

where θ_h is a nonnegative parameter specific to node h and $\lambda_0(t)$ is the baseline hazard. The proportional hazards model is traditionally based on the partial likelihood, however, when the baseline cumulative hazard, $\Delta_0(t)$ is known, estimation and model selection based on the full likelihood are desirable. Therefore, for a tree, H, the full likelihood is expressed as

$$L = \prod_{h \in H} \prod_{i \in \mathcal{L}_h} (\lambda_0(T_i^*)\theta_h)^{\delta_i} \exp^{\Lambda_0(T_i^*)\theta_h}, \tag{2.20}$$

where $\mathcal{L}_h$ is the set of individuals at node h and $\Lambda_0(t)$ is the baseline cumulative hazard [21].

To calculate the likelihood, we need to obtain estimates of the baseline cumulative hazard, $\Lambda_0(T_i^*)$, and the parameter θ_h. When the baseline cumulative hazard is known, the MLE for θ_h is [21]

$$\tilde{\theta}_h = \frac{\sum_{i \in \mathcal{L}_h} \delta_i}{\sum_{i \in \mathcal{L}_h} \Lambda_0(T_i^*)}. \tag{2.21}$$

Unfortunately, in practice the baseline cumulative hazard, Λ_0, is not known. LeBlanc and Crowley [21] estimate it with

$$\hat{\Lambda}_0(t) = \sum_{i: T_i^* \leq t} \frac{\delta_i}{\sum_{h \in H} \sum_{i: T_i^* \geq t \wedge i \in \mathcal{L}_h} \hat{\theta}_h}. \tag{2.22}$$

We can estimate the MLEs of Λ_0 and θ_h iteratively. At iteration j, the cumulative hazard is given by

$$\hat{\Lambda}_0^{(j)}(t) = \sum_{i: T_i^* \leq t} \frac{\delta_i}{\sum_{h \in H} \sum_{i: T_i^* \geq t, i \in \mathcal{L}_h} \hat{\theta}_h^{(j-1)}}, \tag{2.23}$$

where $\hat{\theta}_h^{j-1}$ is the $j-1^{th}$ iteration as an estimate of θ_h. Using the j^{th} iteration estimate of the cumulative hazard, $\hat{\Delta}_0^j(t)$, we can obtain the j^{th} iteration estimate of θ_h [21]

$$\hat{\theta}_h^{(j)} = \frac{\sum_{i \in \mathcal{L}_h} \delta_i}{\sum_{i \in \mathcal{L}_h} \hat{\Lambda}_0^{(j)}(T_i^*)}. \tag{2.24}$$

Typically, an iterative process such as this continues until convergence of the MLEs. However, for efficiency, LeBlanc and Crowley [21] propose to use only the first iteration to get these MLEs. Thus, for iteration $j = 1$, we use $\theta_h^{(0)} = 1$ to estimate $\hat{\Lambda}_h^{(1)}$. The one-step estimate of θ_h is then given by

$$\hat{\theta}_h^{(1)} = \frac{\sum_{i \in \mathcal{L}_h} \delta_i}{\sum_{i \in \mathcal{L}_h} \hat{\Lambda}_0^{(1)}(T_i^*)}, \tag{2.25}$$

and interpreted as the observed number of deaths divided by the expected number of deaths in node h.

LeBlanc and Crowley [21] propose to measure how well the tree fits the data through the full likelihood deviance. For a node h, the deviance is

$$G(h) = 2\{L_h(saturated) - L_h(\tilde{\theta}_h)\}, \tag{2.26}$$

where $L_h(saturated)$ is the log-likelihood for the saturated model that allows one parameter for each observation, and $L_h(\tilde{\theta}_h)$ is the maximized log-likelihood when $\Lambda_0(t)$ is known. For an observation i this is defined as

$$d_i = 2\left[\delta_i log\Big(\frac{\delta_i}{\Lambda_0(T_i^*)\hat{\theta}_h}\Big) - (\delta_i - \Lambda_0(T_i^*)\hat{\theta}_h)\right]. \tag{2.27}$$

To quantify the value a particular split $s_h \in \mathcal{S}_h$ that creates daughter nodes h_L and h_R, LeBlanc and Crowley [21] use the statistic

$$G(s_h) = G(h) - [G(h_l) + G(h_r)], \tag{2.28}$$

where

$$G(h) = \frac{1}{N} \sum_{i \in \mathcal{L}_h} \left[\delta_i log\Big(\frac{\delta_i}{\hat{\Lambda}_0^{(1)}(T_i^*)\hat{\theta}_h^{(1)}}\Big) - (\delta_i - \hat{\Lambda}_0^{(1)}(T_i^*)\hat{\theta}_h^{(1)})\right]. \tag{2.29}$$

The split selected to divide the data is the split that results in the largest decrease in within-node deviance, as defined by equation (2.13).

2.4.4 Pruning to Select the Optimal Subtree

The largest tree created by the growing process, H_{MAX}, can be quite complex when many covariates are used to grow the tree and/or the sample size is large. The tree H_{MAX} actually consists of many subtrees. Subtrees can be created from the main tree by successively pruning off branches, where a branch, H^h, consists of the node h and all its descendants in H. Pruning a branch, H^h, from a tree, H, consists of deleting all descendants of node h from the tree H, that is, removing all of H^h except its root node, h. This pruned subtree is denoted as either $H_h = H - H^h$ to emphasize the missing branch H^h, or $H_h \prec H$ to emphasize the fact that H_h is a subtree of H [18, 26]. Figure 2.2 shows an example of a tree, a branch, and a subtree.

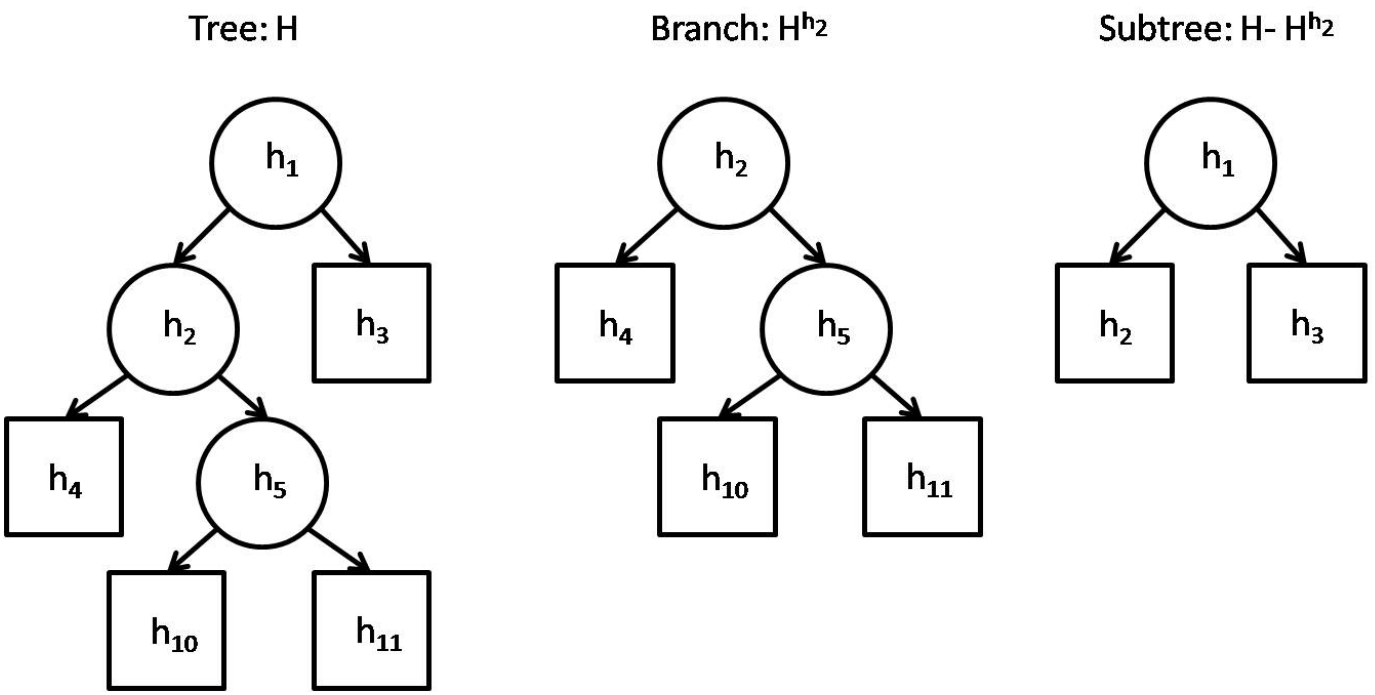

Figure 2.2: Tree pruning

Just as in a regression model, a very large tree model containing every possible covariate may not be clinically meaningful or readily interpretable. Thus, some joint measure of parsimony and accuracy needs to be used to select the best subtree from among all the possible subtrees $H_h \prec H_{MAX}$. The method used to select the best subtree can differ depending on the type of splitting statistic that is used. LeBlanc and Crowley [21] propose a pruning algorithm for their within-node full-likelihood deviance splitting statistic that is very similar to the algorithm described in CART. To assess the value of the prognostic structure of a subtree, H_h, they measure the cost complexity of a subtree, H_h, as

$$G_\alpha(H_h) = \sum_{h \in \tilde{H}_h} G(h) + \alpha |\tilde{H}_h|. \tag{2.30}$$

In this algorithm, $G(h)$ is the full likelihood deviance of a node, h, and α is a nonnegative complexity parameter that can be adjusted to control the penalty for a larger tree.

When the splitting statistic is based on between-node differences, such as the two-sample rank test used by Segal [23] and LeBlanc and Crowley [26], a different pruning algorithm needs to be used. Instead of cost complexity, LeBlanc and Crowley [26] propose using a measure of split complexity to select the best subtree. For any subtree $H_h \prec H_{MAX}$, $I = H_h - \tilde{H}_h$ is the set of internal nodes, that is, all nodes except the terminal nodes $h \in H_h$. They then define the complexity, $|I|$, as the count of the set of internal nodes, I. For a complexity parameter, $\alpha \geq 0$, LeBlanc and Crowley [26] define the split complexity measure, $G_\alpha(H_h)$, as

$$G_\alpha(H_h) = G(H_h) - \alpha |I|, \tag{2.31}$$

where

$$G(H_h) = \sum_{h \in I} G(s_h^*). \tag{2.32}$$

Both the cost-complexity (2.30) and split complexity (2.31) of a subtree $H_h \prec H_{MAX}$ assess the trade-off between overall prognostic ability and tree size. In this dissertation review we focus on split-complexity pruning. Further details regarding cost-complexity pruning for censored survival data can be found in LeBlanc and Crowley [21].

The split-complexity first measures the amount of prognostic structure in the internal nodes of the subtree, H_h, using the sum of the maximum splitting statistics over each node

$h \in I$. It then subtracts a penalty for the size of the subtree. The effect of this penalty is controlled by a complexity parameter, $\alpha \geq 0$. When the chosen α is close to 0, the cost of a large number of internal nodes is small and thus the subtree maximizing the split complexity (2.31) will have more nodes. As the chosen α increases, there will continue to be a higher penalty for subtree complexity, and the subtree maximizing the split complexity (2.31) will have fewer nodes. The subtree consisting of only the root node is associated with the largest α possible [26].

A subtree, H_h, is called an optimally pruned subtree for a selected complexity parameter, α, if

$$G_\alpha(H_h) = \max_{H_h \prec H_{MAX}} G_\alpha(H_h). \tag{2.33}$$

H_h^* is called the smallest optimally pruned subtree if $H_h^* \prec H_h$ for every optimally pruned subtree $H_h \prec H_{MAX}$. One important property, proven by Breiman, et al. [18], for within-node split statistics and adapted by LeBlanc, et al. [26], for between-node split statistics, is that there does indeed exist at least one subtree that maximizes the split-complexity (2.31). As evidenced by this property, a search for an optimally pruned subtree H^* is certain to be successful [18, 26].

With unlimited time and computational power, it would be possible to calculate the split complexity (2.31) of every possible subtree $H_h \prec H_{MAX}$ for any given complexity parameter, α. However, the number of possible subtrees is often large and it would be very computationally inefficient to take this approach. A more effective method of sorting through all the subtrees is called weakest link pruning. This algorithm creates a sequence of nested subtrees by sequentially pruning off the branches that provide the least additional information relative to their size [18, 26]. Note that when weakest link pruning is implemented, H_{MAX} is not used as the starting tree. Instead, the smallest optimally pruned subtree using the complexity parameter $\alpha_1 = 0$ is used as a starting point. This subtree will be called H_1.

The first goal of weakest link pruning is to find the branch providing the least additional information relative to its size. For any nonterminal node $h \in H_1$, consider the branch H_1^h and its split complexity, $G_\alpha(H_1^h)$. As the chosen complexity parameter, α, increases, there will be a threshold for which the value of $\alpha|I_1^h|$ becomes larger than $G(H_1^h)$. At this point, the split-complexity (2.31) for the branch H_1^h becomes negative and it will no longer

be beneficial to split the node h for this value of the parameter, α, because the amount of prognostic structure gained from the split is outweighed by its complexity. This critical value is found by solving the following equality for α

$$G(H_1^h) - \alpha|I_1^h| = 0, \tag{2.34}$$

where $I_1^h = H_1^h - \tilde{H}_1^h$ is the set of internal nodes of the branch and $|I_1^h|$ is the count of the set of internal nodes of the branch. Solving for $\alpha \equiv \alpha(h)$ results in

$$\alpha(h) = \frac{G(H_1^h)}{|I_1^h|} \text{ if } h \in I_1. \tag{2.35}$$

The weakest branch in H_1, denoted $\bar{H}_1^1$, begins at the node, h, that minimizes $\alpha(h)$ for all $h \in H_1$ [26].

For notational ease, let $\alpha_2 = \min_{h \in H_1} \alpha(h)$ and let $H_2 = H_1 - \bar{H}_1{}^1$ be the tree obtained by pruning off branch $\bar{H}^1$ from subtree H_1. Continuing with this notation, let $\alpha_3 = \min_{h \in H_2} \alpha(h)$, and H_3 be the tree obtained by pruning off branch $\bar{H}_2{}^2$ from tree H_2. This sequence continues until all the branches of H_1 have been pruned and only a tree containing the root node, denoted H_R, is remaining. The purpose of this pruning process is to obtain a nested sequence of subtrees $H_R \prec H_{R-1} \prec \ldots \prec H_k \prec H_{k-1} \prec \ldots \prec H_2 \prec H_1$ and the corresponding sequence of parameters $\infty > \alpha_R > \ldots > \alpha_k > \alpha_{k-1} > \ldots > \alpha_2 > \alpha_1 = 0$ [26]. We note that as an α_k increases, it will continue to create the same subtree H_k until the next value, α_{k-1} is reached, at which point the subtree H_{k-1} will be created.

Because a nested sequence of subtrees is available, it now is now computationally feasible to calculate and compare the split complexities (2.31) for each of these nested subtrees. Although it is tempting to simply resubstitute the data used to grow the tree to calculate the split complexity $G_\alpha(H)$ for a tree H, this statistic will be largely overestimated by the sample used to grow the tree H_{MAX} because the split points for each node were chosen based on these data. Therefore, we will discuss two different methods, both proposed by LeBlanc and Crowley [26], for selecting a pruned subtree.

2.4.4.1 Selection of a Subtree through Training and Test Samples

When the sample size is large enough, dividing the sample, $\mathcal{L}$, into a training set, $\mathcal{L}_1$, and testing set, $\mathcal{L}_2$ is recommended [18]. The training set, $\mathcal{L}_1$, is first used to grow the tree and to create a nested sequence of trees through weakest link pruning. The test set, $\mathcal{L}_2$, is sent down each of these subtrees in order to calculate the split-complexity $G_\alpha(H)$ (when between-node splitting statistics are used). The subtree maximizing $G_\alpha(H)$ is chosen as the best tree. Typically, α is selected such that $2 \leq \alpha \leq 4$ if the test statistic is distributed as χ^2_1. Including the penalty term $\alpha|I|$ is necessary, because similar to the R^2 statistic in regression, $G_\alpha(H)$ has the property that it will always increase when more nodes are added to the model.

2.4.4.2 Selection of a Subtree through Bootstrap Resampling

The training/testing set method for subtree selection is computationally efficient, but also requires a large sample size in order to work effectively [26]. Another method, proposed by LeBlanc and Crowley [26] for between-node splitting statistics, is a bootstrap sample method. This method does not require a large N and therefore is often preferred for use with smaller samples.

Let $G(\mathbf{X}_1; \mathbf{X}_2, H) \equiv G(H)$, where $\mathbf{X}_2$ represents the sample used to grow the initial tree, H, and $\mathbf{X}_1$ is sent through the tree, T, to calculate the statistic. LeBlanc and Crowley [26] define the following quantities:

$$\begin{aligned} G^* &= E_F G(\mathbf{X}^*; \mathbf{X}, H) \\ G &= G(\mathbf{X}; \mathbf{X}, T) \\ o &= G^* - G \\ w &= E_F\{G^* - G\}. \end{aligned}$$

If the true distribution of the data, F, were known, then $G(\mathbf{X}; \mathbf{X}, H) + w$ could be used as a bias-corrected $G(H)$, where w represents the over optimism due to utilizing the same sample to grow and test the data. To estimate w in practice, we replace F with the empirical distribution of the training sample, $\mathbf{X}$, and use Monte Carlo techniques to estimate w [26].

To accomplish this, a tree, H, is grown using the entire sample, $\mathcal{L}$. A series of nested subtrees, H_k and their associated α_k values is also created with this same training sample. We then draw B bootstrap samples from $\mathcal{L}$. For each bootstrap sample, $\mathcal{L}^{(b)}$, $b = 1, \ldots, B$,

we grow a tree and find the optimally pruned subtree for each α_k created by the sample $\mathcal{L}$. For each α_k, we calculate

$$o_{b_k} = G(\mathbf{X}; \mathbf{X}_b, H_b(\alpha'_k)) - G(\mathbf{X}_b; \mathbf{X}_b, H_b(\alpha'_k)), \tag{2.36}$$

and take the mean over the B bootstrap samples,

$$\hat{w}_k = \frac{1}{B} \sum_{b=1}^{B} o_{b_k}. \tag{2.37}$$

We choose the tree $H(\alpha'_k)$ that maximizes $\hat{G}_{\alpha_c}(H(\alpha'_k))$, where

$$\hat{G}(H(\alpha'_k)) = G(\mathbf{X}; \mathbf{X}, H(\alpha'_k)) + \hat{w}_k \tag{2.38}$$

and α_c is a previously selected complexity parameter.

2.4.5 Classifying New Individuals

After the tree has been grown and pruned, a new individual can be classified into a terminal node, or prognostic group. When such a new patient is presented, they can be classified into a terminal node based on the existing survival tree algorithm. Since each terminal node is associated with a summary measure of survival, such as the Kaplan Meier product limit estimate or a hazard rate, the survival of the subject based on his or her terminal node can be predicted [34].

2.4.6 Time Dependent Covariates in Tree-Structured Survival Analysis

Tree-structured survival analyses have traditionally utilized fixed covariate information to create prognostic groups. For example, clinical characteristics measured at baseline, as well as descriptive covariates such age and gender are all baseline covariates that can all be used to predict the risk group of a particular individual. There are situations, however, when the value of a covariate at the beginning of a study is expected to change, and can be measured repeatedly over time to get more information. These repeatedly measured covariates could be disease or illness markers such as autoantibody values, CD4 counts, or anxiety scores. Alternatively, these repeatedly measured covariates could be indicators of an interim event, such as surgery or the administration of a new drug. Regardless of the type of repeatedly measured covariate, its changing value over time may provide additional information for the creation of risk groups.

2.4.6.1 Piecewise Exponential Survival Trees with Repeatedly Measured Covariates Huang, Chen, et al. [10] proposed a method for utilizing repeatedly measured covariates for TSSA. Their method splits nodes through the interaction of the covariate values and time, establishing measures of improvement based on the piecewise exponential survival distribution.

The primary emphasis of their method is on modeling the hazard function. Instead of a proportional hazards assumption, however, they model the hazard with a general form $\lambda(t) = \lambda(\mathbf{W}(t), t)$, which simplifies to $\lambda(t) = \lambda(\mathbf{W}, t)$ if all covariates are time independent. When repeatedly measured covariates are used, their method estimates this hazard by a piecewise constant function of time, where the jump points are selected through the tree-structured model. Therefore, it approximates the survival experience of an individual with a piecewise exponential distribution. In the case of time-independent covariates, their method simplifies to the exact method proposed by Davis and Anderson [20].

The algorithm approximates the distribution of an event time, T_i, by a piecewise expo-

nential distribution of k pieces, which has a density function

$$f_i(t) = \begin{cases} \lambda_{i_1} \exp(-\lambda_{i1} t), & 0 = t_{i_0} < t \leq t_{i1} \\ \lambda_{i_2} \exp(t_{i_1}(\lambda_{i_2} - \lambda_{i_1}) - \lambda_{i_2} t), & t_{i_1} < t \leq t_{i_2} \\ \vdots & \\ \lambda_{i_k} \exp\left[\sum_{j=1}^{k-1} t_{i_j}(\lambda_{i_{j+1}} - \lambda_{i_j} - \lambda_{i_k} t\right], & t_{i_{k-1}} < t \leq t_{i_k}, \end{cases} \tag{2.39}$$

where $0 = t_{i_0} < t_{i_1} < \ldots < t_{i_k} = \infty$ and $\lambda_{i_1}, \lambda_{i_2}, \ldots, \lambda_{i_k}$ are positive.

For simplicity, we describe their methodology for the case of one repeatedly measured covariate nondecreasing in time. If the root node splits on a value of c_0 on covariate $W(t)$, a subject, i, with time-dependent observations $W_i(t_{ij}) = [w_i(t_{i1}), \ldots, w_i(t_{iJ_i})]$ can belong to one of three categories

1. $W_i(t_{ij}) \leq c$ at all time points goes to the left daughter node
2. $W_i(t_{ij}) > c$ at all time points goes to the right daughter node
3. $W_i(t_{ij}) \leq c$ (at times $0 < t \leq t_i^*$) contributes to the left daughter node and $W(t) > c$ (at times $t_i^* < t \leq T_i^*$) contributes to the right daughter node, where t_i^* is the last time point for which $W_i(t_{ij}) \leq c$

The algorithm is handled analogously for nonincreasing repeatedly measured covariates, and can be extended to include nonmonotonically changing covariates.

Huang et al. [10] provide an example of a survival tree constructed with a repeatedly measured indicator variable for transplant status. Furthermore, they performed simulations to assess the piecewise exponential survival tree model in comparison to the Cox proportional hazards model. Of particular concern was whether their proposed model, which creates step functions for changing hazards over time, could accommodate continuously changing hazards. Three different settings were used to investigate their model: (1) survival times dependent on only a time-independent covariate, (2) survival times related to only a repeatedly measured covariate, and (3) survival times related to both repeatedly measured and time-independent covariates. The repeatedly measured covariate used in the model was nondecreasing monotonically with either two, three, or four categories (corresponding to one, two, or three possible split points).

To assess their model and compare it to the Cox model, the true relative risks were compared to the estimated mean relative risks using the mean square error and its standard deviation. Results showed that the categorical repeatedly measured covariates in the piecewise exponential model performed either as well or slightly better than the Cox model when the survival times were related to either the time-independent or time-dependent covariate. However, the Cox model performed better than their piecewise exponential tree model when the survival times were dependent on both a time-independent and a time-dependent covariate.

2.4.6.2 Time-Dependent Survival Trees with the Two-Sample Rank Statistic

Bacchetti and Segal [9] use the two-sample rank statistic to accommodate time-dependent covariates in tree-structured survival analysis. Similar to Huang et al. [10], subjects are divided into "pseudo-subjects" when the value of their time-dependent covariate is below the selected cut point at some time points and above the selected cut point at other time points. They apply their methodology to create a tree model for time to HIV seroconversion, using calendar time as a time-dependent covariate.

Bacchetti and Segal formally explain the creation of pseudo-subjects as follows. Consider an individual, i, $i = 1, \ldots, N$ followed from his/her time of entry, τ_i, to his/her outcome time, T_i^*, with time-dependent covariate $W_i(t_{ij}) = [w_i(t_{i1}), \ldots, w_i(t_{iJ_i})]$. For a selected split, c, subjects i with $W_i(t_{ij})) > c$ at all times are sent to the right node, while subjects i with $W_i(t_{ij}) \leq c$ at all times are sent to the left node. However, there may also exist subjects whose time-dependent covariate $W_i(t_{ij})$ is greater than c at some time points and less than or equal to c at other time points. Such subjects need to contribute to the left node at some times and the right node at other times.

To illustrate how this occurs, Bacchetti and Segal present the case where $W_i(t_{ij})$ is nondecreasing in time, and denote t_i^* as the last time when $W_i(t_{ij}) \leq c$, with $\tau_i < t_{ij}^* < T_i^*$. To properly test the potential split c, subject i must be considered part of the left node for times t_{ij} such that $\tau_i < t_{ij} \leq t_i^*$ and part of the right node at failure times $t_i^* < t_{ij} \leq T_i^*$. Such a scenario is easily incorporated when left-truncation is accommodated. Specifically, we consider the i^{th} subject's survival experience as being composed of two non-overlapping

survival experiences of two pseudo-subjects, i_1 and i_2. Pseudo-subject i_1 is at risk only up to time t_i^*, at which point they are right censored. Pseudo-subject i_2 is left truncated at t_i^* and is at risk until their event time, T_i^*. Pseudo-subjects i_1 and i_2 are assigned to the right and left nodes, respectively.

The two-sample test for a potential split, c, is calculated as in equation 7.1. This statistic can accommodate pseudo-subjects because an individual can only be assigned to either the left or right node at any given time point. When the log rank statistic is used, the test is equivalent to fitting a univariate proportional hazards model with a time-dependent covariate equal to the indicator $1\{W_i(t_{ij}) \leq c\}$. If $W_i(t_{ij})$ is monotonically decreasing in time, the split is handled analogously with t_i^* now defined as the last time when $W_i(t_{ij}) > c$.

Bacchetti and Segal also propose two ways to generalize this method non-monotonically increasing or decreasing covariates, examples of which are shown in Figure 2.3. These methods do not differ in their calculation of the splitting statistic, but do differ in the way they assign pseudo-subjects after the split has been selected. The first method proposed is to split an individual, i, into two or more pseudo-subjects, each representing a segment of time during which $W_(t_{ij}) \leq c$ or $W_i(t_{ij})) > c$. For example, suppose $W_i(t_{ij}) \leq c$ for $t_{ij} \leq t_i^{*1}$ and $t_{ij} > t_i^{*2}$, but $W_i(t_{ij}) > c$ for $t_i^{*1} < t_{ij} \leq t_i^{*2}$. To split this observation, we would assign three sets of pseudo-observations: $(\tau_{i_1}, T_{i_1}^*, \delta_{i_1}) = (\tau_i, t_i^{*1}, 0)$, $(\tau_{i_2}, T_{i_2}^*, \delta_{i_2}) = (t_i^{*2}, t_i^{*1}, 0)$, $(\tau_{i_3}, T_{i_3}^*, \delta_{i_3}) = (t_i^{*2}, T_i^*, \delta_i)$.

Creating more than two pseudo-subjects can become relatively complex Therefore, Bacchetti and Segal also suggest a second method for creating pseudo-subjects with nonmonotonically increasing or decreasing time-dependent covariates. This method restricts an individual, i, to only two pseudo-subjects by creating indicator variables that equal one if subject i was at risk at the j^{th} distinct failure time and zero otherwise. Pseudo-subject i_1 is assigned to those time points for which the indicator variable is equal to one, and pseudo-subject i_2 is assigned to those time points for which the indicator variable is equal to zero.

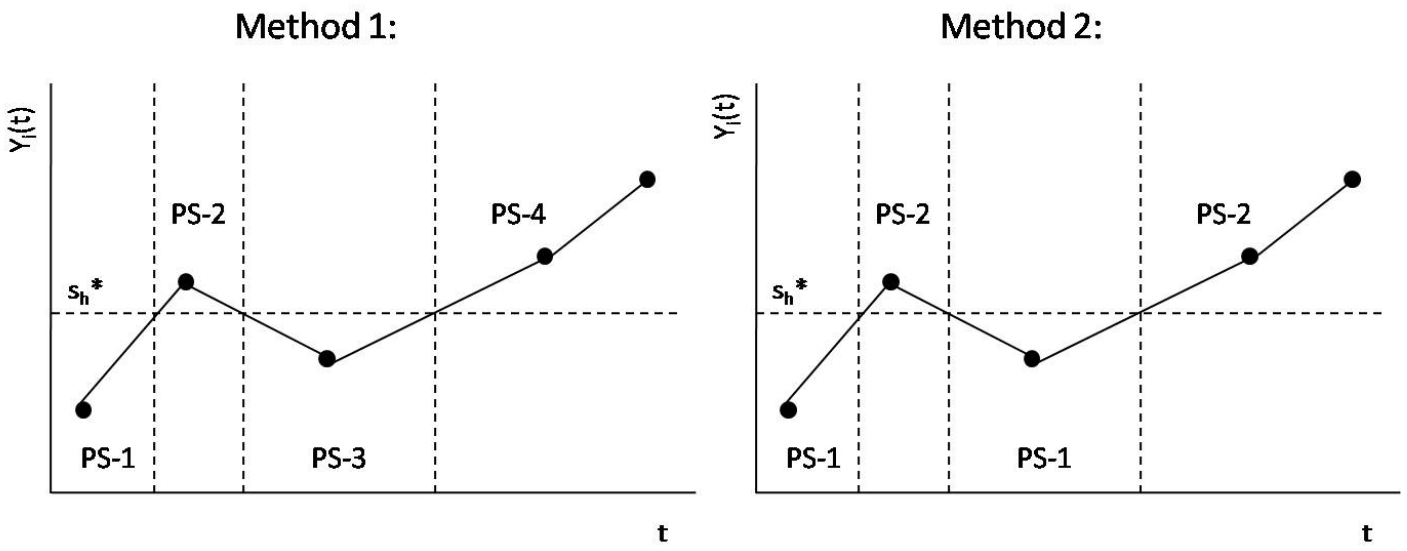

Figure 2.3: Two methods for creating pseudo-subjects

2.4.7 Tree Stability

Many researchers have assessed the effectiveness of tree-based modeling, subsequently proposing new methods to make the trees more stable and less biased. For example, not long after CART was released, Crawford [35] suggested bootstrap extensions to the standard CART algorithm to improve its pruning abilities. Later, Dannegger [36] addressed many issues related to tree-structured survival analysis, such as tree stability, factor importance, and multiple testing. In addition, he assessed current bootstrap techniques, and from his results created updated techniques to enhance the performance of the trees. Hothorn et al. [37] also assessed the performance of survival trees, recommending a bagging algorithm to make the survival trees more stable.

2.5 SURVIVAL ANALYSIS DIAGNOSTICS

Assessing the accuracy of the predictions obtained from a survival analysis is useful because it provides a measure of certainty of the model. Two associated measures used to assess the accuracy of a prognostic model are the Brier Score [8] and the mean integrated squared error (MIE) [37]. One of the many advantages of these statistics is that estimated probabilities are used to predict the event status at an event time, t_j. Thus, predictions are made in terms of the probability an individual is classified into a given prognostic category, as opposed to simply classifying the individual as having an event or having no event.

The Brier score for censored data at an event time, t_j, $j = 1, \ldots, J$, is given by

$$BS(t_j) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1(T_i^* \leq t_j)\hat{S}(t_j^*|X_i)^2\delta_i}{\hat{G}(T_i^*)} + \frac{1(T_i^* > t_j^*)\{1-\hat{S}(t_j^*|X_i)\}^2}{\hat{G}(t_j)}\right], \quad (2.40)$$

where $\hat{G}(T_i^*)$ denotes the estimate of the censoring distribution, $G(T_i^*)$, and $\hat{S}(T_i^*)$ denotes the estimate of the survival distribution, $S(T_i^*)$. The contributions to the Brier score can be split up into three categories

1. $T_i^* \leq t_j$ and $\delta_i = 1$,
2. $T_i^* > t_j$ and $\{\delta_i = 1$ or $\delta_i = 0\}$, and

3. $T_i^* \leq t_j$ and $\delta_i = 0$.

Category 1 contains individuals i who have an event before t_j. Their contribution to the Brier score is $\frac{\hat{S}(t_j|X_i)^2}{\hat{G}(T_i^*)}$. Category 2 contains individuals i who either have an event or are censored after time t_j. Their contribution to the Brier score is $\frac{\{1-\hat{S}(t_j|X_i)\}^2}{\hat{G}(t_j)}$. For the individuals in category 3 who were censored before t_j, the event status at t_j is unknown and therefore their contribution to the Brier score cannot be calculated However, these individuals do contribute the weight $1/N$ [8].

The individual contributions need to be re-weighted to compensate for the loss of information due to censoring. Thus, the individuals in category 1 who only survive until time $T_i^* < t_j$ are weighted by $\hat{G}(T_i^*)^{-1}$ and the individuals in category 2 who survive until t_j are weighted by $\hat{G}(t_j)^{-1}$. Although the individuals from category 3 do not contribute their event free probabilities to $\hat{S}(t_j^*|X_i)$, they do contribute to the weights $\{N\hat{G}(T_i^*)\}^{-1}$ and $\{N\hat{G}(t_j)\}^{-1}$.

When repeatedly measured covariates are employed, we incorporate modifications presented by Schoop et al. [38] to estimate the Brier score. At time t_j, the empirical Brier score with repeatedly measured covariates is estimated by

$$BS(t_j) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1(T_i^* \leq t_j)\hat{S}(t_j; W_i^{t_j})\delta_i}{\hat{G}(T_i^*)} + \frac{1(T_i^* > t_j)\{1-\hat{S}(t_j; W_i^{t_j})\}}{\hat{G}(t_j)}\right], \quad (2.41)$$

where $W_i^{t_j} = \{W_i(t_{ij}), t_{i1} \leq t_{ij} \leq t_j\}$ denotes individual i's covariate history up to time t_j, $\hat{S}(t_j; W_i^{t_j})$ estimates the survival distribution at time t_j conditional on covariate information available up to t_j, and $\hat{G}$ estimates the censoring distribution.

The Brier score incorporating censored data (2.40) can be calculated at all time points t_j, $j = 1, \ldots, J$, separately. However, it can also be integrated with respect to a weight function, w_j,

$$IBS = \int_{j=1}^{J} BS(t_j) dw_j. \quad (2.42)$$

Natural choices for the weight function are $w_j = t_j/t_J$ or $w_j = \{1-\hat{S}(t_j)\}/\{1-\hat{S}(t_J)\}$, where $\hat{S}(t_j)$ denotes the observed rate of event-free patients at time t_j. When discrete events

times are used, the integrated Brier score can be rewritten as

$$IBS = \sum_{j=1}^{J} \frac{1}{w_j} BS(t_j), \tag{2.43}$$

where J represents the number of unique event times.

For both the fixed time and repeatedly measured covariate scenarios, the hypothetical case of perfect foresight results in a Brier score equal to zero, that is, no difference between prediction and true failure status. A Brier score of .25 corresponds to the trivial rule of always assigning 50% probability, and therefore is the upper boundary of a reasonable prediction. A value of one is the maximum Brier score, corresponding to the case of perfect "inverse" foresight.

3.0 LITERATURE REVIEW: MISSING COVARIATE DATA

3.1 MISSING DATA MECHANISMS

The missing data mechanism describes the relationship between the missing and observed values in a dataset. To formally describe this mechanism, consider the complete data matrix $\mathbf{X}_{n\times p} = \{x_{ik}\}$ and missing-data indicator matrix $\mathbf{M}_{n\times p} = \{m_{ik}\}$, where $m_{ik} = 1$ when x_{ik} is missing and $m_{ik} = 0$ when x_{ik} is observed. Missing data mechanisms are characterized by the conditional distribution $f(M|X, \phi)$, where ϕ represents unknown parameters [3].

There are three main types of missing data mechanisms: missing completely at random (MCAR), missing at random (MAR), and not missing at random (NMAR). The missing data mechanism is considered MCAR when the missingness does not depend on either the observed or unobserved values of $X = (X_{obs}, X_{mis})$, such that

$$f(M|X, \phi) = f(M|\phi) \text{ for all } X,\ \phi, \tag{3.1}$$

where ϕ represents nuisance parameters. The presence of the MCAR mechanism implies that the observed cases are a completely random subsample of all the cases (both observed and unobserved) [3]. The presence of the MAR mechanism implies that the missingness depends on the observed data (X_{obs}) and not the missing data (X_{mis}) that is,

$$f(M|X, \phi) = f(M|X_{obs}, X_{mis}, \phi) = f(M|X_{obs}, \phi) \text{ for all } X_{mis},\ \phi. \tag{3.2}$$

The MAR mechanism is much less stringent than MCAR and thus is easier to assume in practice. The presence of the NMAR mechanism implies that the conditional distribution of M depends on the missing values in X, that is,

$$f(M|X, \phi) = f(M|X_{obs}, X_{mis}, \phi) = f(M|X_{mis}, \phi) \text{ for all } X_{mis},\ \phi. \tag{3.3}$$

The missing-data indicator, M, may also depend on all or part of X_{obs} in addition to X_{mis}, but it is not necessary a condition for the data to be NMAR [3].

3.2 HANDLING MISSING COVARIATE DATA

3.2.1 General Strategies

3.2.1.1 Completely Observed and Available Case Analyses The simplest approaches to handling missing data are the "completely observed" and "available" analysis methods. The completely observed method analysis confines attention to cases where all the variables are present, allowing for standard analytical procedures to be used. Alternatively, the available case method utilizes all cases as long as the variable of interest is present [3].

Although the methods are simple to use, both the completely observed and available case analyses make no use of cases missing a covariate X_k when estimating either the marginal distribution of X_k or measures of covariation between X_k and other variables. This may lead to a significant decrease in sample size and subsequently a loss of power It may lead to biased results if the data are not MCAR.

3.2.1.2 Regression-based Single Imputation One way the information from the cases missing covariate X_k can be recovered is by singly imputing the values of missing items. After single imputation, a complete case analysis with the full sample can be used to obtain more accurate estimates without a loss of power.

There are numerous imputation models that can be used to impute missing values. Little and Rubin [3] describe four general types of explicit imputation models: unconditional mean, unconditional draw, conditional mean, and conditional draw. The two unconditional methods do not utilize any covariate information to impute the data. We will focus on the somewhat more sophisticated imputation techniques of conditional mean (Cmean) and conditional draw (Cdraw). For the purpose of explaining the Cmean and Cdraw imputation methods, we consider bivariate data with X_1 fully observed and X_2 partially observed, such

that the first r cases of X_2 are observed and the last $n-r$ cases of X_2 are missing.

Cmean imputation utilizes information from nonmissing covariates to more effectively estimate the missing covariate information. The first step for imputation using the Cmean technique is to regress X_2 on X_1 using only the first r nonmissing observations, that is

$$X_2^* = \beta_0 + \beta_1 X_1^* + \epsilon,$$

where $X_2^* = [x_{12}, x_{22}, \ldots x_{r2}]'$ and $X_1^* = [x_{12}, x_{22}, \ldots, x_{r2}]'$. Fitting this equation provides estimates of the intercept and slope parameters, $\hat{\beta}_0$ and $\hat{\beta}_1$. Next, each x_{i1} value, $i = r+1, \ldots, n$ is sequentially plugged into the regression equation

$$\hat{x}_{i2} = \hat{\beta}_{00} + x_{i1}\hat{\beta}_1. \tag{3.4}$$

Each predicted value, $\hat{x}_{i2}$, $i = r, \ldots, n$, is imputed into its corresponding missing value in covariate X_2.

Imputation using the Cdraw technique is similar to the Cmean technique. However, instead of imputing only the predicted value into the dataset, it imputes the predicted value plus a randomly drawn normal deviate unique to each individual missing X_2. This random normal deviate is distributed as $\epsilon_i \sim N(0, \tilde{\sigma}_2)$, where $\tilde{\sigma}_2$ is the sample variance of X_2 based on the $i = 1, \ldots, r$ completely observed cases. After adding this randomly drawn normal deviate to equation 3.2.1.2, the estimate for an individual, i, is

$$\hat{x}_{i2} = \hat{\beta}_0 + x_{i1}\hat{\beta}_1 + \epsilon_i. \tag{3.5}$$

Just as in the Cmean imputation technique, each predicted value, $\hat{x}_{i2}$, $i = r+1, \ldots, n$ is imputed into its corresponding missing value in covariate X_2.

Cdraw is the preferred single imputation method of Little and Rubin because it yields consistent estimates of the mean, the standard deviation, and the regression estimates. This property of the Cdraw imputation method holds for both MCAR and MAR data [3]. Of course, there are drawbacks to the Cdraw method. First, there is a loss of efficiency due to the random draws. Secondly, the standard errors of the parameter estimates from the filled-in data are too small because they do not incorporate imputation uncertainty. However, this second drawback is a problem with both the conditional and unconditional imputation

methods described by Little and Rubin. Regardless of the single method used, additional uncertainty due to nonresponse should be added into the estimates of parameter variance.

3.2.1.3 Tree-Based Single Imputation The Cmean methodology proposed by Little and Rubin [3] utilizes linear regression to model the observed data and subsequently impute missing data. Alternatively, Conversano and Siciliano [5] proposed a Cmean single imputation method that uses tree-structured regression to model the observed data. For this method, consider a covariate matrix, $\mathbf{X}_{n\times p}$, containing r cases with completely observed values and $n-r$ cases without an observed X_k value. As shown in Figure 3.1, the matrix $\mathbf{X}_{n\times p}$ can be separated into four submatrices, where $\mathbf{A}$ and $\mathbf{B}$ contain data for the completely observed cases, $\mathbf{C}$ contains the observed data coinciding with the cases missing X_k, and $\mathbf{D}$ represents the missing X_k data.

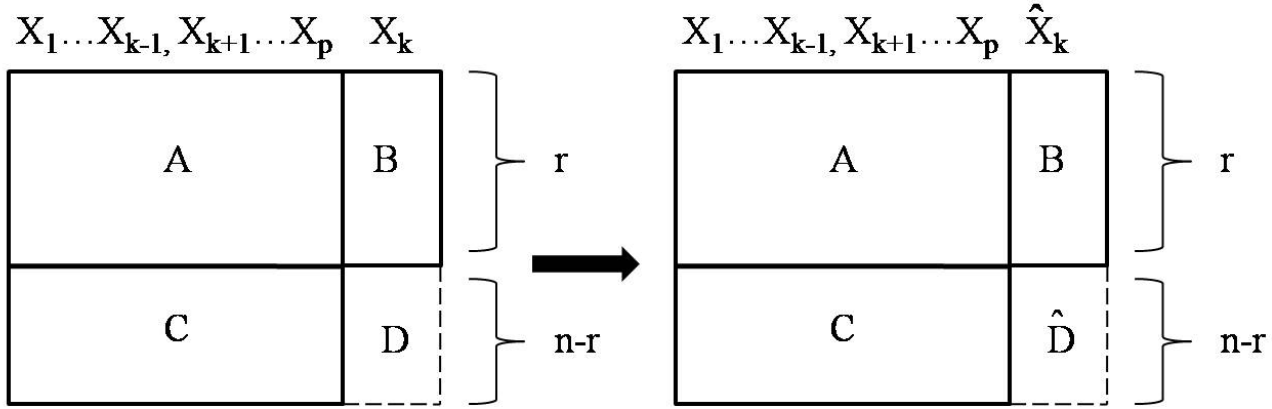

Figure 3.1: Data structure before and after single tree-based imputation

A regression tree, H_k, is grown using the observed X_k values in $\mathbf{B}$ as the outcome and the corresponding covariate data in $\mathbf{A}$ as the predictor space. For each terminal node $h \in H_k$, we calculate the mean of the observed X_k data,

$$\bar{X}_{kh} = \frac{1}{r_h} \sum_{i \in \mathcal{L}_h} X_{ik}, \tag{3.6}$$

where $\mathcal{L}_h$ is the set of r_h individuals in terminal node h, with $\sum_{h \in H_k} r_h = r$. The tree model, H_k, is then used to classify the cases in $\mathbf{C}$ and $\mathbf{D}$ into a terminal node, h based on their non-missing covariate data in $\mathbf{C}$. For cases classified into a terminal node, h, the value $\bar{X}_{kh}$ is imputed into the corresponding row of $\mathbf{D}$. After imputation, the set of $n - r$ imputed mean values is denoted $\hat{\mathbf{D}}$ and the covariate is denoted $\hat{X}_k$, as shown in Figure 3.1 [5].

3.2.1.4 Multiple Imputation

Little and Rubin discuss multiple methods to add uncertainty due to nonresponse. However, they assert that creating multiply imputed data sets yields the least biased standard error estimates without compromising the parameter estimates themselves [3].

When using multiple imputation, we replace each missing value with a vector of $M \geq 2$ imputed values, and M completed datasets are created from these vectors. The m^{th} dataset is created by replacing each missing value by the m^{th} value in each of the corresponding imputation vectors. After creating multiple datasets, standard complete data methods are used to estimate a parameter value for each set. Little and Rubin [3] note that when the M sets of imputations are repeated random draws from the predictive distribution of missing values, the M inferences can be combined to form one inference that properly reflects uncertainty due to nonresponse.

3.2.2 Methods for Tree-Structured Models

3.2.2.1 Surrogate Splitting

To our knowledge, surrogate splitting is the most well documented method for handling missing covariates in trees, largely due to the fact that it is utilized in both the CART and RPART algorithms [18, 4, 29]. To formally describe this method, consider a covariate, X_k, with r_k observed values and $n - r_k$ missing values. When

determining the best split point on the covariate X_k, only the r_k individuals with observed X_k are utilized. If a split, s_h^* on covariate X_k is selected to divide the sample, all n individuals need to be sent to either the left or right node to continue to be in the tree model. The node destinations are clear for the r_k observed values. For example, in the continuous case, observations with $x_{ik} \leq s_h^*$ go to the left daughter node, h_L, and observations with $x_{ik} > s_h^*$ go to the right node, h_R. However, due to the missing values, there exist $n - r_k$ observations without a clear node destination.

Surrogate splitting assumes that there is another split on a covariate X_r, $r = 1, \ldots, k-1, k+1, \ldots, p$, that can take the place of the split s_h^* on covariate X_k to classify individuals into the nodes created by s_h^*. To determine the best surrogate splitting value, we use the sample of r_k individuals with observed X_k data to detect the single split, s_r, on covariate X_r that best classifies each individuals into the node that was assigned to them based on the originally selected splitting value, s_h^*. The resulting misclassification error rate for a surrogate covariate X_r is denoted

$$m_r = (\#\text{misclassified}/r_k),$$

and is used to rank each possible surrogate covariate X_r, $r \neq k$.

One exception to this ranking procedure is when a surrogate variable, X_r, has a misclassification rate, m_r, that is worse than a blind "go with the majority" misclassification rate. This blind rate for X_r is calculated as $min(p_r, 1 - p_r)$, where $p_r = (\# \leq s_r)/N_h$ and N_h represents the number of observations in the node h. When this is the case, the blind rate, p_r is used to rank the surrogate variable instead of the misclassification rate, m_r. The surrogate variable with best ranking is used to send the $n - r_k$ cases missing X_k to either the left or right node, based on whether the observed value of the surrogate variable is above or below the selected cut point.

If one of the $n - r_k$ individual is also missing the selected surrogate variable, the next best ranked surrogate variable is used, and so on. If an observation is missing all the surrogate variables, the observations are sent to the daughter node holding the most observations.

3.2.2.2 Multiple Imputation with Bagging

An alternative to surrogate splitting is to impute the missing observations. For example, Feelders [2] used Bayesian methodology to create a multiply imputed dataset and grew a classification tree based on each imputation set. After each model was created, bagging techniques, as proposed by Hothorn et al. [37], were used to classify new cases based on the multiple tree models. The bagging methodology used the multiple trees to create an overall "best" classification for each new case under consideration. Simulation study results showed this bagging method to have a lower classification error rate than either its single imputation counterpart or surrogate split methodology.

4.0 MAINTENANCE THERAPIES IN LATE-LIFE DEPRESSION

Our methodology and subsequent applications are based on the Maintenance Therapies in late-life Depression (MTLD) clinical trial [1]. This clinical trial was conducted to determine the effectiveness of nortriptyline hydrochloride and interpersonal therapy (IPT) in preventing recurrence of major depressive episodes. Participants in the MTLD-I clinical trial were required to be older than 59 and to meet expert clinical judgment and diagnostic criteria for recurrent, nonpsychotic, nondysthymic, unipolar major depression. After a baseline screening, the study began with an acute treatment phase that could last up to 26 weeks. In this phase, 180 patients received open treatment using a combination of nortriptyline hydrochloride with plasma steady-state levels of 80-120 ng/L and weekly psychotherapy. After remission from depression they were observed in a continuation phase, and as long as they did not relapse they were randomized to maintenance therapies with a 2×2 randomized block design including placebo drug vs nortriptyline and a medical clinic vs IPT.

Findings from the original MTLD study were focused on the effectiveness of the maintenance therapies, discussed by Reynolds et al. [1]. A Kaplan Meier analysis with the log-rank statistic was used to test for differences in the time to recurrence of depression in the four randomized maintenance therapy treatment groups. The results of this analysis showed a significant effect for active treatment over placebo in preventing recurrence of major depressive episodes, with the best outcome seen in patients assigned to the combined IPT/nortriptyline treatment condition. A Cox model was also used to test and control for the effects of age at study entry, number of prior episodes of major depression, duration of acute therapy, social support (Interpersonal Support Evaluation List), and chronic medical burden (Cumulative Illness Rating Scale for Geriatrics). Higher age at study entry was associated with a greater probability of recurrence; none of the other clinical covariates were significant.

Table 4.1: Characteristics at the start of acute treatment phase

Covariate (Scale*)	N	Mean(SD)	Min.	Med.	Max.
Depression (HRS-D)	151	19.14(4.07)	7	19	27
Anxiety (BSI-Anx)	151	1.41(0.94)	0	1.33	3.83
Cumulative Illness (CIRS)	151	7.22(3.29)	1	7	19
Self Esteem (ISEL-SE)	151	4.56(3.08)	0	4	12
Sleep Quality (PSQI)	114	11.06(4.19)	2	11.5	20
Age	151	67.33(5.82)	59	67	91
Depression Duration (yrs.)	151	19.03(16.35)	0	14	63
Nortriptyline ng/mL (Week 2)	148	48.62(31.15)	2	42.5	151

*HRS-D: Hamilton 17-Item Depression Rating Scale, BSI-Anx: Brief Symptom Inventory Anxiety Subscale, CIRS: Cumulative Illness Rating Scale, ISEL-SE: Interpersonal Support Evaluation List Score Self Esteem Subscale, PSQI: Global Pittsburgh Sleep Quality Index

The main outcome paper by Reynolds et al. [1] analyzed data from the maintenance phase; however, our primary interest is in the acute phase. Specifically, we desired to use the data from the acute phase to determine clinical characteristics that put individuals at higher risk for not responding to treatment to depression. We use a sample of N=151 individuals followed during the acute phase of the MTLD-I clinical trial to motivate and illustrate our methodology. These individuals had baseline depression, anxiety, CIR, and ISEL-SE, as well as at least three consecutive observed depression observations during follow-up. The event of interest, treatment response during acute treatment, was defined as occurring when an individual's 17-item Hamilton Depression Rating Scale (HRSD) dropped ≤ 10 for at least three consecutive weeks. Clinical characteristics of this sample at the start of acute treatment are shown in Table 4.1.

Although our primary methodological interest in this dissertation is TSSA, we wanted the reader to get an initial sense for the relationships between the covariates of interest and the time to treatment response. Therefore, we use the Cox model for preliminary univariate

and multivariate models. Results of the Cox model including only fixed time covariates are shown in Table 4.2. Unit increases in HRS-D, BSI-anx, CIR, PSQI, and duration of depression are all associated with lower rates of treatment response. Alternatively, unit increases in age and self-esteem are associated with higher rates of treatment response. The covariates with $p \leq .1$ in the univariate Cox models were put into a multivariate Cox model. Results are shown in model nine in Table 4.2, with unit increases in week one depression score significantly associated with lower rates of treatment response.

In addition to the covariate measurements at week one, some covariates, such as anxiety, were also measured weekly. Plots of anxiety scores over time are shown in Figures 4.1 and 4.2. At each week, the individuals who had treatment had a mean anxiety lower than those individuals without a treatment response at almost every time point, and those time points for which this was not true had very low sample sizes.

Table 4.2: Baseline Cox model results

Model	Covariate	Hazard Ratio	SE	p-value
1	**HRS-D (week 1)**	.914	.023	<.0001
2	**BSI-Anx (week 1)**	.707	.112	.002
3	**CIR (week 1)**	.989	.030	.700
4	**ISEL-SE (week 1)**	1.060	.031	.060
5	**PSQI (week 1)**	.937	.027	.015
6	**Age (week 1)**	1.013	.017	.448
7	**Depression Duration (yrs.) (week 1)**	.991	.006	.152
8	**Nortriptyline ng/mL (week 2)**	1.004	.003	.197
9	**Multivariate:**			
	PSQI (week 1)	.960	.028	.150
	BSI-Anx (week 1)	.9	.134	.433
	HRS-D (week 1)	.921	.034	.016
	ISEL-SE (week 1)	1.045	0.039	.266

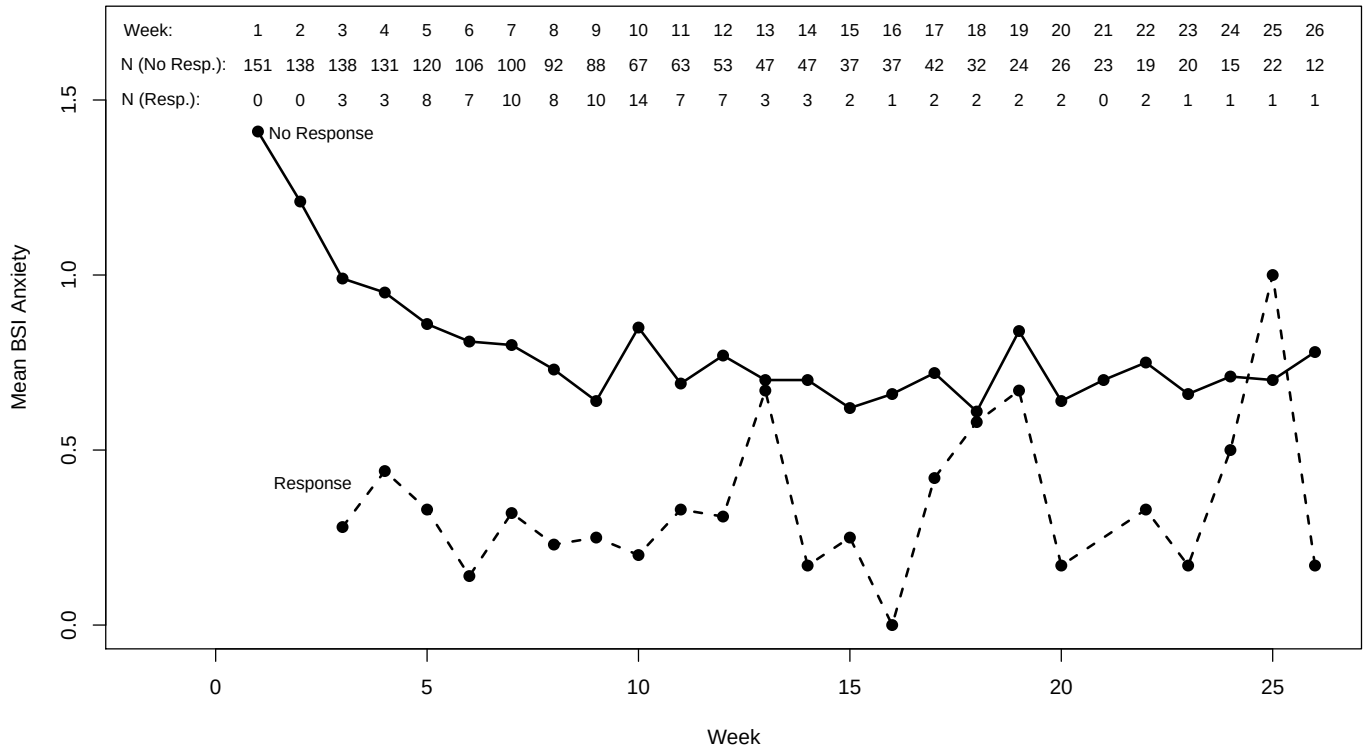

Figure 4.1: Mean anxiety at each week of acute treatment, by treatment response

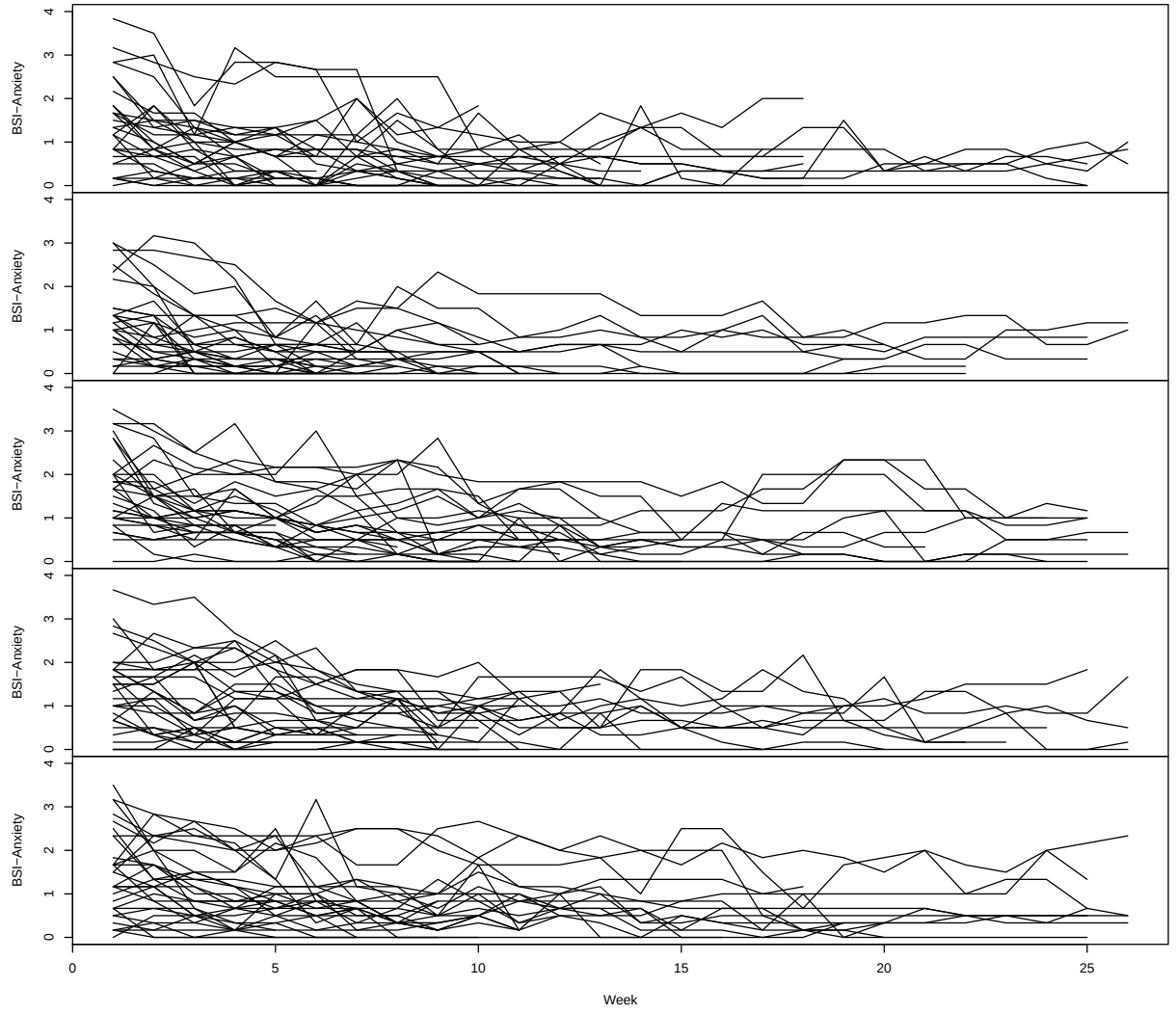

Figure 4.2: Individual anxiety at each week of acute treatment

Table 4.3: Time-Dependent Cox model results

Model	Covariate	Hazard Ratio	SE	p-value
1	**Time-Dependent BSI-Anx**	.182	.259	<.0001
2	**Multivariate:**			
	Time-Dependent BSI-Anx	.274	.286	<.0001
	PSQI (week 1)	.965	.029	.211
	HRS-D (week 1)	.945	.032	.076
	ISEL-SE (week 1)	1.019	.038	.618

We also fit the Cox model with anxiety analyzed as a repeatedly measured covariate, as shown in Table 4.3. Using only baseline covariates, HDRS was the sole significant predictor ($p = .016$) in the multivariate Cox model. However, when time-dependent anxiety was substituted for baseline anxiety, it replaced HDRS as the only significant predictor ($p < .0001$). These results suggest that the additional information gleaned from the updated anxiety measurements at each time point are extremely helpful in predicting the outcome. Individuals with a one-unit higher BSI-anxiety level were .247 times more likely to respond to treatment for depression, or alternatively 4.05 times more likely to not respond to treatment.

5.0 A STOCHASTIC MULTIPLE IMPUTATION ALGORITHM FOR MISSING COVARIATE DATA

When imputing covariates that will be used as predictors in tree-structured analysis, it is important to represent the missing data as accurately as possible so that the correct split point is available to be selected and the observations can be classified into the correct node. Conversano and Siciliano's tree-structured single imputation method [5], described in Section 3.2.1.3 can only impute as many unique values as there are terminal nodes in the tree. Although this may be appropriate for covariates with a limited number of categorical values, we do not view this as an effective imputation strategy for covariates with a larger set of possible values because it is not likely to realistically represent the missing observations. With this in mind, we propose a new tree-based multiple imputation strategy, where draws of stochastic error are added to the imputation estimates proposed by Conversano and Siciliano.

5.1 METHODOLOGY

5.1.1 Creating Multiple Imputation Estimates

Similar to the single imputation method by Conversano and Siciliano, we begin with an $\mathbf{X}_{n\times p} = \{x_{ik}\}$ data matrix containing $n - r$ missing $\mathbf{X}_k$ observations and divided into four submatrices as shown in the left side of Figure 5.1. Also, as in Conversano and Siciliano's method, a tree, H_k, is grown using $\mathbf{A}$ as the predictor space and $\mathbf{B}$ as the outcome, and the mean value $\bar{X}_{kh}$ in each terminal node, $h \in H_k$, is calculated. Unlike the single imputation method, however, we now also calculate a set of observed residuals, R_{kh}, for each terminal

node as follows

$$R_{kh} = \{x_{ik} - \bar{X}_{kh} : i \in \mathcal{L}_h\}, . \tag{5.1}$$

where $\mathcal{L}_h$ is the set of individuals in node h. The regression tree, H_k, is used to classify the $n - r$ cases in $\mathbf{C}$ and $\mathbf{D}$ into terminal nodes based on their corresponding observed data in $\mathbf{C}$.

For a case classified into terminal node $h \in \tilde{H}_k$, we create M imputation estimates

$$\hat{x}_{ik}^{(m)} = \bar{X}_{kh} + \epsilon_{ih}^{(m)}, \ m = 1, \ldots, M, \tag{5.2}$$

where each $\epsilon_{ih}^{(m)}$ is assumed to be distributed normally with a mean of zero and a variance equal to the variance of the observed residuals in terminal node h, as follows

$$\epsilon_{ih}^{(m)} \sim N\Big(0, \widehat{Var}(R_{kh})\Big). \tag{5.3}$$

After creating multiple imputation estimates for each case, i, in submatrix $\mathbf{D}$, we obtain M imputed vectors of length $n - r$. Each of these vectors is denoted $\hat{\mathbf{D}} + \boldsymbol{\epsilon}^{(m)}$, where $\hat{\mathbf{D}}$ represents the imputed vector of terminal node means from Conversano and Siciliano's method and $\boldsymbol{\epsilon}^{(m)}$ represents one of the M vectors of stochastic error. Each imputed vector, $\hat{\mathbf{D}} + \boldsymbol{\epsilon}^{(m)}$, is appended to a replicate of submatrix $\mathbf{B}$ to create M vectors of length n, denoted $\hat{\mathbf{X}}_k^{(m)}$, $m = 1, \ldots, M$. The data structure after multiple imputation is shown in the right side of Figure 5.1.

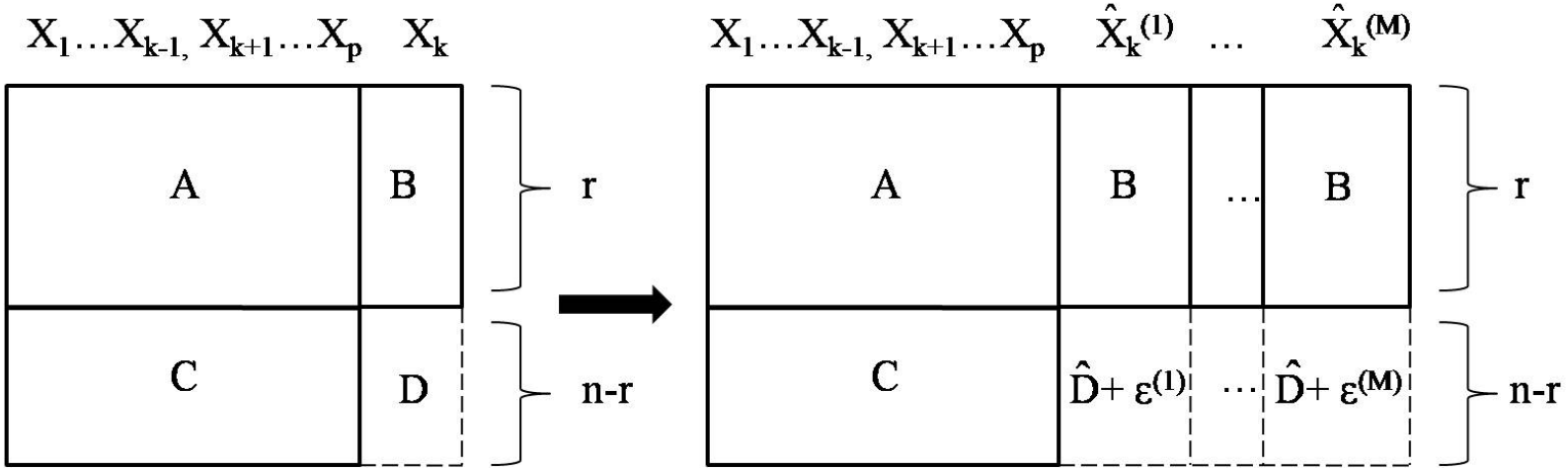

Figure 5.1: Data structure before and after multiple imputation

5.1.2 Incorporating Multiple Imputations Into a Single Tree Model

When selecting the best split at a node, h, we need to consider splits on the imputed covariate and also the set of non-imputed covariates. In general, we accomplish this by first selecting the best split from the non-imputed covariates $\mathbf{X}_j$, $j \neq k$, and then comparing it to each of the M vectors representing covariate $\mathbf{X}_k$.

Formally, we first identify the best split, s_h^*, and associated statistic, G_h^*, based on only the non-imputed covariates X_j, $j \neq k$. Next, we identify the best split, $s_h^{*(m)}$, and associated statistic, $G(s_h^{*(m)})$, for each imputed covariate vector $\hat{\mathbf{X}}_k^{(m)}$, $m = 1, \ldots, M$. To find the overall best split, s_h^{**}, each $G(s_h^*)^{(m)}$, $m = 1, \ldots, M$, is compared to the best statistic from the non-imputed covariates, $G(s_h^*)$, as follows

$$s_h^{**} = \begin{cases} s_h^{(*)} & \text{if } \sum_{m=1}^{M} \{I(G(s_h^{*(m)}) > G(s_h^*))\} \geq M/2 \\ s_h^* & \text{if } \sum_{m=1}^{M} \{I(G(s_h^{*(m)}) > G(s_h^*))\} < M/2 \end{cases}, \tag{5.4}$$

where

$$s_h^{(*)} = \text{Median}(\{s_h^{*(m)} : m = 1, \ldots, M\}). \tag{5.5}$$

The value $M/2$ in equation 5.4 corresponds to selecting a split on the covariate X_k when it is the best split for at least half of the multiple imputation vectors.

If the best split, s_h^{**}, occurs at one of the $\mathbf{X}_j$, $j \neq k$, non-imputed covariates, the tree-growing process can proceed as usual and all observations are sent to either the left node, h_L, or right node, h_R, depending on whether their $\mathbf{X}_j$ value is less than or equal to s_h^{**} or greater than s_h^{**}, respectively. However, if the multiply imputed covariate, $\mathbf{X}_k$, is chosen, the r cases with observed $\mathbf{X}_k$ can be sent to a node based on that value, but the $n - r$ observations missing covariate $\mathbf{X}_k$ will still need a node assignment. For individual, i, missing value x_{ik}, a node assignment, h_i, is made as follows

$$h_i = \begin{cases} h_L & \text{if } a_i \geq M/2 \\ h_R & \text{if } a_i < M/2 \end{cases}, \tag{5.6}$$

where

$$a_i = \sum_{m=1}^{M} I(X_{ik}^{(m)} \leq s_h^{**}). \tag{5.7}$$

The value $M/2$ corresponds to sending the case i to the left node when it is less than or equal to the split value, s_h^{**}, for at least half of the multiple imputation vectors.

This algorithm continues iteratively at each node further down the tree. If there are multiple covariates with missing data, the process can be extended so that each of the M imputed vectors for each covariate with missing data are compared to the complete data values to select the best split.

After growing the largest tree possible, H_{MAX}, weakest link pruning can be implemented by utilizing the statistic

$$G_h^{**} = \begin{cases} G(s_h^{(*)}) & \text{if } \sum_{m=1}^{M}\{I(G(s_h^{*(m)}) > G(s_h^*))\} \geq M/2 \\ G(s_h^*) & \text{if } \sum_{m=1}^{M}\{I(G(s_h^{*(m)}) > G(s_h^*))\} < M/2 \end{cases}, \tag{5.8}$$

where

$$G(s_h^{(*)}) = Median\{G(s_h^{(m)}) : m = 1, \ldots, M\}. \tag{5.9}$$

To save computational time, we suggest using a value of $M = 1$ to impute any missing observations in the test dataset(s), and proceed with weakest link pruning as detailed by Breiman et al. [18] or LeBlanc and Crowley [21, 26].

5.2 SIMULATION STUDY

The goal of this simulation study was to assess the ability of our multiple imputation algorithm to create accurate tree-structured survival models in comparison to other currently used methods. We assessed the accuracy of our method with three percentages of MAR covariate data, (10%, 25%, and 40%), two splitting statistics (full likelihood deviance [21] and two-sample log rank [23, 26]), and two complexity levels between covariates (denoted model A and model B).

For each combination of percentage missing, splitting statistic and complexity level, our strategy was to first generate a "complete" dataset and then remove a proportion of the observations in one covariate. We refer to the data not removed as the "observed" dataset, and to the subset of the observed dataset that contains only the cases with completely

Table 5.1: Simulated data structure for models A and B

	Model A	Model B
Model Covariates	$V_1 \sim \text{N}(10, 2)$	$V_1 \sim \text{N}(\mu_h, 3)$*
	$V_2 \sim \text{Bin}(.6)$	$V_2 \sim \text{Bin}(.6)$
	$V_3 \sim \text{N}(13 + .5V_1, 4)$	$V_3 \sim \text{N}(25, 6)$
	$V_4 \sim \text{Bin}(.5)$	$V_4 \sim \text{Bin}(.5)$
Extraneous Variables	$V_5 \sim \text{N}(20, 5)$	$V_5 \sim \text{N}(20, 5)$
	$V_6 \sim \text{Bin}(.7)$	$V_6 \sim \text{Bin}(.7)$
	$V_7 \sim \text{Bin}(.5)$	$V_7 \sim \text{Bin}(.5)$
	$V_8 \sim \text{Bin}(.4)$	$V_8 \sim \text{Bin}(.4)$
Missingness Indicators for Covariate V_1	$M_1 \sim \text{Bin}(.2I(V_3 < 18))$	$M_1 \sim \text{Bin}(.2I(V_3 < 25))$
	$M_2 \sim \text{Bin}(.5I(V_3 < 18))$	$M_2 \sim \text{Bin}(.5I(V_3 < 25))$
	$M_3 \sim \text{Bin}(.8I(V_3 < 18))$	$M_3 \sim \text{Bin}(.8I(V_3 < 25))$
Event Time p.d.f.	$\text{Exp}(\lambda_h)^\dagger$	$\text{Exp}(\lambda_h)^\dagger$
Censoring Time p.d.f.	$f(c) = .1$, if $c = 1/4, 1/2, 1, 2$; $= .2$, if $c = 4, 8, 16$	$f(c) = .1$, if $c = 1/4, 1/2, 1, 2$; $= .2$, if $c = 4, 8, 16$

*See Figure 5.2.1 for values of μ_h in covariate V_1

†See Figure 5.2.1 for values of λ_h, for both models A and B

observed data as the "completely observed" dataset. Based on these datasets, we compared five different methods:

(1) Using the completely observed dataset to grow trees (COBS),

(2) Using surrogate splits to grow trees with the observed dataset (SS),

(3) Imputing the observed dataset with Conversano and Siciliano's method and using the resulting dataset to grow trees (CS),

(4) Imputing the observed dataset using our multiple imputation method with $M = 1$ and using the resulting dataset to grow trees (MI-1), and

(5) Imputing the observed dataset and growing trees using our multiple imputation algorithm with $M = 10$ (MI-10).

For comparison purposes, we also grew a trees using the complete dataset (COMP) before removing a proportion of the data to create missing observations.

5.2.1 Data Generation

We generated continuous and binary covariates to be used in the true survival tree models as well as extraneous continuous and binary covariates to be used as noise. Model A had a limited structure between covariates such that only two covariates were correlated. Model B had a more complex structure between covariates, such that the mean value of covariate V_1 for each observation was determined by the tree algorithm in Figure 5.2.1. For both models A and B, we generated three missing data indicators that would allow us to create datasets with approximately 10%, 25%, and 40% of observations missing at random in covariate V_1.

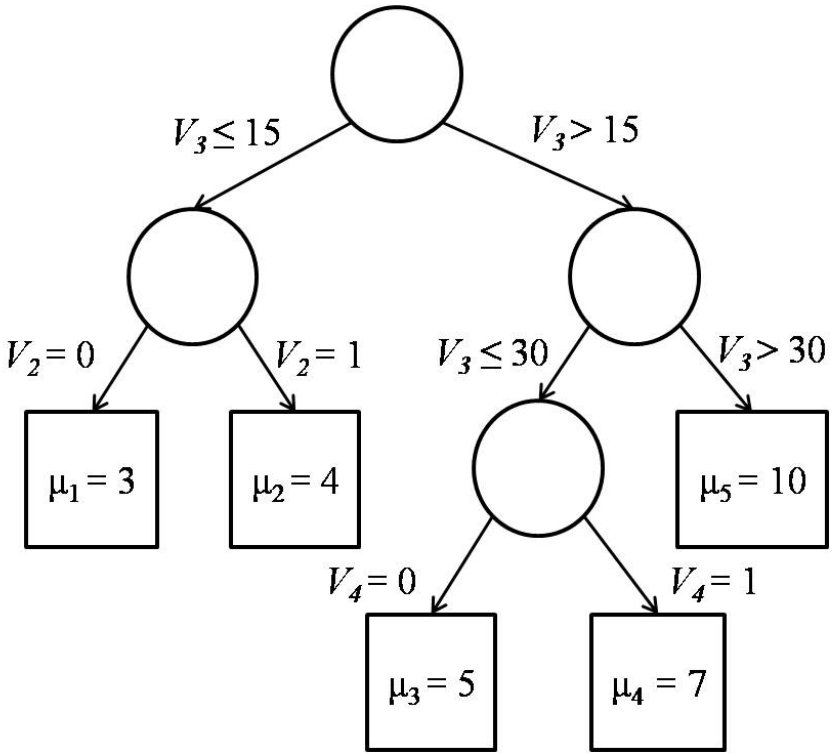

Figure 5.2: Parameter values for covariate V_1 in model B

The distributions of the true event times, T, were exponential with the differing parameters specified as indicated in the tree model with five terminal nodes, shown in Figure 5.2.1. We arranged this tree model so that the covariate with missing data (V_1) split node h_1 and the covariate associated with the missingness (V_3) split below it at node h_3. To reflect a practical application, we generated discrete censoring times with a lower probability at

earlier time points ($Pr(C = c) = .1$ for times $c = .25, .5, 1, 2$) and a higher probability at later time points ($Pr(C = c) = .2$ for times $c = .25, .5, 1, 2$). The observed outcome time was taken to be the minimum of the event and censoring distributions.

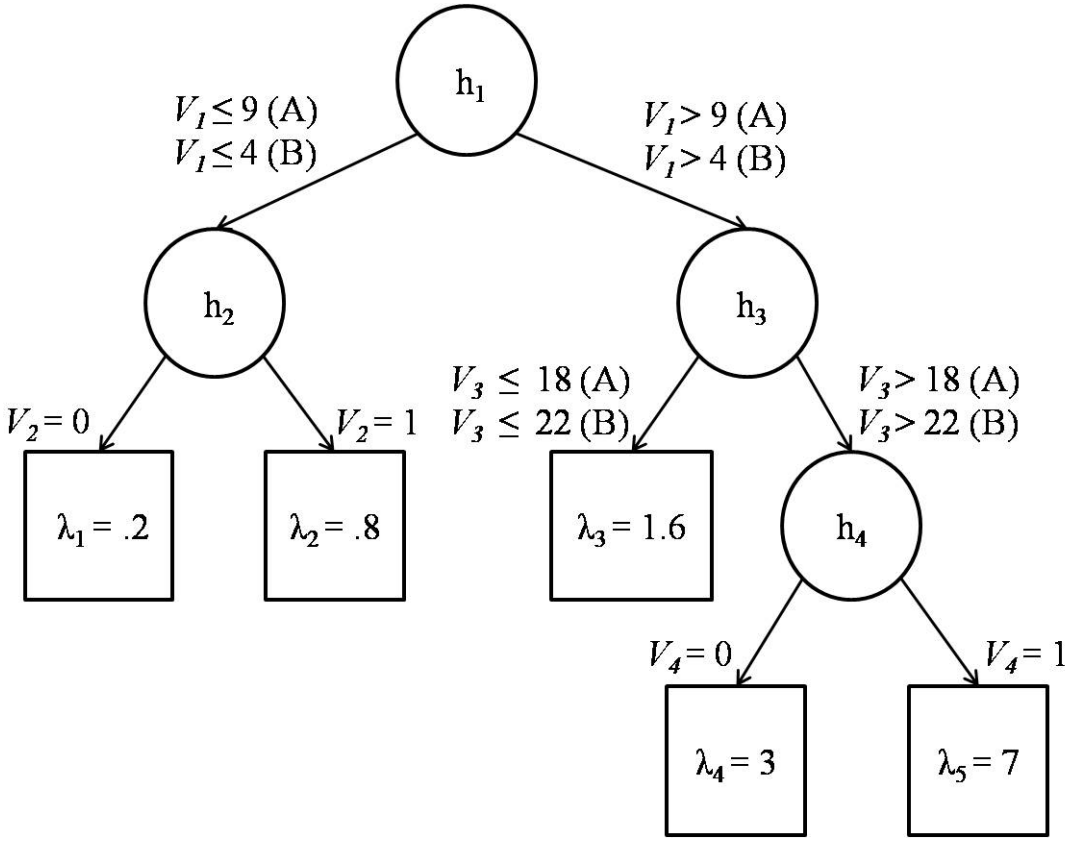

Figure 5.3: Parameter values for the true event time

Altogether, we simulated 1000 datasets of N=300 observations for model A and another 1000 datasets for model B. The 1000 datasets created for model A had, on average, 16.15% censored observations and missingness indicators to create datasets with 9.95%, 24.93%, and 40.07% missing observations in covariate V_1. The 1000 datasets created for model B had, on average, 17.82% censored observations and missingness indicators to create 10.05%, 24.94%, and 40.08% missing observations in covariate V_1. The statistical program R [30] was used for all data generation.

5.2.2 Computing Methods

We utilized the "rpart()" function with the "method=anova" option in R to create the single and multiple tree-structured regression imputation models (CS, MI-1, and MI-10) [4, 29]. In these imputation models, the observed values in covariate V_1 were used as the outcome and all other covariates, $V_2 - V_8$, were used as possible predictors. We required at least 20 observations to split a node and at least ten observations in each terminal node. Other than these specifications, the trees were grown and pruned using the standard parameters in the "rpart()" function.

We wrote functions in R (see Appendix A.1) to create the survival trees for all five missing data methods [30]. For the trees grown with the full likelihood deviance (FLD) splitting statistic, the function "rpart()" specifying the option "class=exp" was utilized within our function to determine the best split at each node [4, 29]. We wrote our own function in R (see Appendix A.1) to find the best split for a node based on the log rank (LR) statistic because *rpart* does not incorporate this statistic. The surrogate split algorithms for both splitting statistics were based on work by Therneau, et al. [4, 29] and Breiman et al. [18]. For all survival models, we allowed a node to split if it held at least 20 observations, and a split was only considered if it would result in at least ten observations per daughter node. This resulted in some lower nodes not being able to split due to not enough observations being sent down the tree. To focus specifically on the trees' abilities to detect the correct splits at nodes $h_1 - h_4$, we did not utilize any pruning techniques.

5.2.3 Assessment Measures

We assessed the methods for handling missing covariate data in TSSA based on four accuracy criteria

(1) The percentage of models that split on the correct covariate at each node.

(2) The percentage of models that split on any of the noise variables $V_5 - V_8$ at each node.

(3) The mean bias and empirical standard deviation (ESD) of the splitting statistic at nodes h_1 and h_3 (which split on continuous covariates).

(4) The percentage of models that selected the correct covariates at all four nodes.

For criteria (1) and (2), the percentage correct was conditional on the model reaching the designated node of interest through the correct covariates, and denominator values were adjusted accordingly. For criterion (3), the mean and ESD were taken only from those models that correctly reached the node of interest and then correctly selected the covariate to split that node. For criteria (1) through (3), our main interests were nodes h_1 and h_3 because the covariate that split node h_3 (V_3) was related to the missingness in the covariate that split node h_1 (V_1).

5.2.4 Model A Results

Criterion 1: With only 10% missing, all methods were comparably accurate at selecting the correct covariate at nodes h_1 and h_3. With 25% and 40% missing, the imputation methods (CS, MI-1, and MI-10) were less accuate than the COBS and SS methods at node h_1. Conversely, at node h_3, the imputation methods were more accurate than the COBS and SS methods. This was the case for both the FLD and LR statistics, although the effect was more extreme with the LR statistic. Overall, the MI-10 method was more accurate than the MI-1 method at both nodes h_1 and h_3.

Criterion 2: None of the methods split on the noise covariates at node h_1. At node h_3, the MI-10 and MI-1 methods selected noise covariates less often than the CS, OBS, and SS methods.

Criterion 3: At node h_1, the bias and ESD of the splitting values were lowest for the COBS and SS methods. The MI-1 and MI-10 methods had the next lowest biases and ESDs, followed by the CS method. At node h_3, the MI-1 and MI-10 methods had the lower biases and ESDs than the other three methods. Interestingly, the biases resulting from the FLD statistic were almost all negative, whereas the biases resulting from the LR statistic were all positive.

Criterion 4: As shown in Figure 5.4, the overall abilities of the COBS, SS, CS, and MI-10 methods to create an accurate tree were comparable to the abilities of the COMP method with only 10% missing. At 25% missing, only the CS method dropped in overall accuracy compared to the other methods. At 40% missing, the MI-10 method was more accurate than

the other methds, and the CS method continued to be the least accurate. Regardless of the method used to manage the missing observations, the FLD statistic consistently created a higher percentage of completely accurate trees compared to the LR statistic, particularly for higher percentages of missing data. The MI-1 results are not shown in Figure 5.4 due to their similarity to the MI-10 results, but are included in Table II.

Table 5.2: Simulation results from model A with full likelihood deviance statistic

		Node h_1 True Split: V_1		Node h_2 True Split: V_2		Node h_3 True Split: V_3			Node h_4 True Split: V_4			Nodes $h_1 - h_4$
Statistic* (% Missing)	Method**	% Corr.†	Bias (ESD)‡	% Corr.	% $V_5 - V_8$	% Corr	% $V_5 - V_8$	Bias (ESD)	N from h_1 & h_3§	% Corr.	% $V_5 - V_8$	% Corr.
FLD (0%)	COMP	100	.01(.09)	97.60	.60	99.90	0	-.01(.46)	999	91.69	2.80	89.3
FLD (10%)	COBS	100	.01(.10)	95.40	1.80	99.30	0	-.03(.54)	993	91.64	3.52	86.5
	SS	100	.01(.10)	94.90	1.40	99.70	0	-.03(.47)	997	92.08	3.21	87.1
	CS	99.80	.04(.17)	94.29	1.30	99.80	0	-.01(.39)	996	92.07	2.51	86.5
	MI-1	99.70	.02(.12)	95.89	1.30	100	0	-.02(.36)	997	92.08	2.91	87.9
	MI-10	99.90	.01(.09)	95.90	.90	99.90	0	-.01(.41)	998	92.18	2.81	87.9
FLD (25%)	COBS	100	.01(.11)	90.70	3.2	93.70	.4	-.08(.64)	937	90.72	4.06	77.4
	SS	99.80	.01(.10)	84.57	3.21	97.29	.30	-.06(.56)	971	91.56	3.50	74.8
	CS	97.30	.16(.36)	79.96	2.36	99.08	.10	-.02(.43)	964	89.42	3.63	69.9
	MI-1	95.70	.04(.17)	86.65	3.23	99.90	0	-.05(.34)	958	91.54	3.03	75.6
	MI-10	97.30	.02(.10)	85.30	3.39	99.90	0	-.02(.30)	972	92.29	3.29	76.2
FLD (40%)	COBS	100	.00(.10)	79.50	8.40	60.40	1.10	-.12(1.08)	604	88.08	4.97	41.7
	SS	99.00	.00(.09)	69.90	3.43	72.83	.61	-.11(.85)	721	89.04	5.13	43.9
	CS	90.90	.53(.69)	53.14	3.85	82.51	1.87	.02(.76)	750	84.53	6.13	38.3
	MI-1	82.30	.10(.31)	73.03	4.86	99.88	0	-.03(.32)	822	90.88	2.68	54.6
	MI-10	87.10	.05(.18)	67.16	5.63	99.77	0	-.03(.27)	869	92.06	3.11	53.7

*FLD: Full likelihood deviance, LR: Two-Sample Log Rank

**COMP: Full simulated data, COBS: completely observed cases only, SS: surrogate split methodology, CS: Conversano and Siciliano's single conditional mean tree-structured imputation method, MI-1: Multiple imputation method with only one draw, MI-10: Multiple imputation method with ten draws.

† % Corr: Percentage of models that selected the correct covariate to split the node.

‡ Bias (ESD): Mean bias and empirical standard deviation of the splitting values for the models that selected the correct covariate to split the node.

§ Number of models that selected the correct covariates at node h_1 and node h_3. Used as the denominator value when calculating the percentage of models that selected the correct covariate at node h_4.

Table 5.3: Simulation results from model A with log rank statistic

		Node h_1 True Split: V_1		Node h_2 True Split: V_2		Node h_3 True Split: V_3			Node h_4 True Split: V_4			Nodes $h_1 - h_4$
Statistic* (% Missing)	Method**	% Corr.†	Bias (ESD)‡	% Corr.	% $V_5 - V_8$	% Corr	% $V_5 - V_8$	Bias (ESD)	N from h_1 & h_3§	% Corr.	% $V_5 - V_8$	% Corr.
LR (0%)	COMP	99.90	.03(.12)	95.70	1.30	99.90	0	.10(.67)	998	90.68	3.41	86.4
LR (10%)	COBS	99.90	.03(.13)	93.29	1.90	99.00	.10	.09(.72)	989	90.39	4.04	83.3
	SS	99.80	.03(.13)	93.89	1.40	99.40	.10	.08(.68)	992	90.73	3.83	84.7
	CS	99.20	.11(.27)	88.41	1.61	99.40	.50	.10(.64)	986	89.45	3.96	79.4
	MI-1	99.50	.04(.14)	93.27	2.11	99.80	0	.08(.60)	993	90.94	3.73	84.1
	MI-10	99.50	.03(.12)	94.17	1.31	99.90	0	.09(.61)	994	90.74	3.52	84.9
LR (25%)	COBS	100	.03(.13)	90.00	3.1	92.80	.50	.07(.96)	928	89.22	4.42	74.6
	SS	99.60	.03(.14)	84.94	2.71	96.69	.20	.06(.78)	963	90.34	3.84	73.7
	CS	94.80	.36(.57)	68.57	2.64	96.20	.63	.06(.58)	912	87.17	5.26	58.2
	MI-1	93.80	.11(.31)	85.29	2.88	99.89	0	.03(.46)	937	90.18	3.63	72.3
	MI-10	94.80	.05(.15)	82.17	3.06	99.89	0	.02(.37)	947	91.13	3.48	70.6
LR (40%)	COBS	99.90	.02(.12)	79.38	7.81	53.45	2.50	.15(1.31)	534	83.15	6.74	35.0
	SS	97.70	.02(.12)	72.26	3.99	67.35	1.64	.07(1.08)	658	86.62	5.47	41.4
	CS	90.70	.89(.87)	43.66	3.20	68.47	5.95	.24(1.02)	621	81.00	6.28	28.3
	MI-1	75.10	.24(.57)	66.31	6.26	98.27	.53	.02(.39)	738	88.89	4.88	44.7
	MI-10	80.00	.15(.36)	62.63	6.13	97.75	.14	.04(.37)	782	91.56	3.71	45.4

*FLD: Full likelihood deviance, LR: Two-Sample Log Rank

**COMP: Full simulated data, COBS: completely observed cases only, SS: surrogate split methodology, CS: Conversano and Siciliano's single conditional mean tree-structured imputation method, MI-1: Multiple imputation method with only one draw, MI-10: Multiple imputation method with ten draws.

† % Corr: Percentage of models that selected the correct covariate to split the node.

‡ Bias (ESD): Mean bias and empirical standard deviation of the splitting values for the models that selected the correct covariate to split the node.

§ Number of models that selected the correct covariates at node h_1 and node h_3. Used as the denominator value when calculating the percentage of models that selected the correct covariate at node h_4.

5.2.5 Model B Results

Criterion 1: With only 10% missing, all methods were comparably accurate at selecting the correct covarate at node h_1. With larger percentages of missing data, the imputation methods (CS, MI-1, and MI-10) were less accurate than the COBS and SS methods at node h_1. At node h_3, however, the imputation methods were much more accurate than the COBS and SS methods, which showed an extreme decrease in accuracy from their results in model A.

Criterion 2: None of the methods split on noise covariates at node h_1. At node h_3, the MI-10 and MI-1 methods split on fewer noise covariates than the other methods. In general, all methods split on more noise covariates with larger percentages of missing data and when using the LR splitting statistic.

Criterion 3: The three imputation methods had larger biases and ESDs than the COBS and SS methods at nodes h_1 and h_3. The bias of the split values for all methods typically increased as the missingness percentage increased. Furthermore, the LR statistic generally was associated with larger biases and ESDs.

Criterion 4: As shown in Figure 5.4, the MI-10 method outperformed the COBS, SS, and CS methods at each percentage of missingness. The SS method consistently had the lowest percentage of trees that split on all the correct covariates, showing an extremely large decrease in accuracy from the results in model A. The MI-1 and MI-10 methods were quite close in overall model accuracy, with the MI-10 method outperforming the MI-1 method by only a few percentage points for each set of simulations. The MI-1 results for criterion (4) are not shown in Figure 5.4 due to their similarity to the MI-10 results, but are shown in Table III.

Table 5.4: Simulation results from model B with full likelihood deviance statistic

		Node h_1 True Split: V_1		Node h_2 True Split: V_2		Node h_3 True Split: V_3			Node h_4 True Split: V_4			Nodes $h_1 - h_4$
Statistic* (% Missing)	Method**	% Corr.†	Bias (ESD)‡	% Corr.	% $V_5 - V_8$	% Corr	% $V_5 - V_8$	Bias (ESD)	N from h_1 & h_3§	% Corr.	% $V_5 - V_8$	% Corr.
FLD (0%)	COMP	100	.01(.09)	83.80	6.70	97.60	0	-.04(.58)	976	98.16	.51	80.3
FLD (10%)	COBS	100	.01(.13)	79.10	7.50	93.70	0	-.01(.84)	937	97.44	.75	72.1
	SS	100	.01(.13)	57.00	11.90	93.90	0	.02(.84)	939	97.44	.75	51.5
	CS	100	.05(.22)	74.80	8.20	98.00	0	.31(1.04)	980	97.35	.82	71.3
	MI-1	99.90	.03(.16)	73.37	9.81	97.60	0	.24(.96)	975	97.74	.82	69.6
	MI-10	100	.01(.13)	76.20	8.50	97.80	0	.29(1.00)	978	97.96	.82	72.7
FLD (25%)	COBS	100	.02(.14)	66.70	12.20	80.10	.40	.00(1.19)	801	96.63	.50	51.7
	SS	99.70	.02(.14)	27.88	8.93	82.65	.30	.06(1.23)	824	96.84	.49	22.1
	CS	98.10	.26(.61)	50.25	10.19	96.63	0	.74(1.19)	948	94.20	1.79	45.8
	MI-1	97.60	.08(.32)	55.33	11.48	98.67	0	.72(1.20)	963	96.16	1.45	51.6
	MI-10	98.70	.04(.17)	56.74	11.55	98.28	0	.78(1.16)	970	96.62	1.13	53.9
FLD (40%)	COBS	100	.02(.18)	51.40	19.00	37.30	1.80	.51(2.28)	373	91.15	1.88	17.7
	SS	97.80	.01(.16)	11.96	3.68	39.47	1.74	.56(2.19)	386	91.71	2.07	5.1
	CS	92.50	.67(1.03)	26.92	8.54	88.54	.86	.95(1.43)	819	90.96	2.69	22.5
	MI-1	83.30	.23(.64)	29.65	10.80	99.28	.12	1.07(1.32)	827	95.16	1.81	23.7
	MI-10	87.60	.12(.32)	29.79	10.73	98.63	0	1.19(1.31)	864	95.14	1.85	24.9

*FLD: Full likelihood deviance, LR: Log Rank

**COMP: Full simulated data, COBS: completely observed cases only, SS: surrogate split methodology, CS: Conversano and Siciliano's single conditional mean tree-structured imputation method, MI-1: Multiple imputation method with only one draw, MI-10: Multiple imputation method with ten draws.

† % Corr: Percentage of models that selected the correct covariate to split the node.

‡ Bias (ESD): Mean bias and empirical standard deviation of the splitting values for the models that selected the correct covariate to split the node.

§ Number of models that selected the correct covariates at node h_1 and node h_3. Used as the denominator value when calculating the percentage of models that selected the correct covariate at node h_4.

Table 5.5: Simulation results from model B with log rank statistic

		Node h_1 True Split: V_1		Node h_2 True Split: V_2		Node h_3 True Split: V_3			Node h_4 True Split: V_4			Nodes $h_1 - h_4$
Statistic* (% Missing)	Method**	% Corr.†	Bias (ESD)‡	% Corr.	% $V_5 - V_8$	% Corr	% $V_5 - V_8$	Bias (ESD)	N from h_1 & h_3§	% Corr.	% $V_5 - V_8$	% Corr.
LR (0%)	COMP	100	.05(.15)	86.40	5.30	96.80	0	.10(.91)	968	97.31	1.14	81.5
LR (10%)	COBS	100	.05(.17)	82.60	5.70	91.80	.10	.12(1.06)	918	96.41	1.63	73
	SS	100	.05(.17)	64.90	10.8	92.00	.10	.14(1.06)	920	96.85	1.30	57.8
	CS	100	.20(.43)	71.60	5.30	97.10	.10	.53(1.32)	971	95.57	2.27	66.4
	MI-1	99.90	.09(.24)	78.18	6.90	96.80	.10	.44(1.13)	967	97.0	1.45	72.7
	MI-10	100	.06(.19)	78.40	7.40	97.50	0	.55(1.24)	975	96.62	1.64	73.6
LR (25%)	COBS	100	.07(.20)	68.9	11.0	76.20	.80	.22(1.55)	762	95.80	1.05	50.5
	SS	99.30	.07(.19)	35.55	10.17	78.55	.70	.33(1.65)	780	95.51	1.15	25.8
	CS	96.40	.66(.92)	41.49	5.91	93.88	.83	.96(1.49)	905	93.48	2.20	38.3
	MI-1	94.20	.24(.59)	57.64	11.04	97.66	.11	1.11(1.39)	920	95.43	1.63	51.6
	MI-10	96.00	.12(.26)	58.75	7.60	98.02	0	1.14(1.30)	941	95.54	1.70	52.9
LR (40%)	COBS	100	.07(.22)	38.0	24.3	28.60	2.80	1.53(3.19)	286	83.92	2.10	8.7
	SS	96.00	.07(.22)	19.90	5.10	31.25	2.92	1.32(2.81)	300	86.33	2.00	5.8
	CS	89.30	1.45(1.40)	18.14	5.45	72.79	4.14	1.25(2.08)	650	87.23	3.84	13.3
	MI-1	74.20	.56(1.03)	34.50	9.97	91.71	.26	1.40(1.38)	719	94.16	2.36	23.8
	MI-10	78.60	.38(.67)	34.35	8.14	96.69	.64	1.50(1.25)	760	93.55	2.24	24.8

*FLD: Full likelihood deviance, LR: Log Rank

**COMP: Full simulated data, COBS: completely observed cases only, SS: surrogate split methodology, CS: Conversano and Siciliano's single conditional mean tree-structured imputation method, MI-1: Multiple imputation method with only one draw, MI-10: Multiple imputation method with ten draws.

† % Corr: Percentage of models that selected the correct covariate to split the node.

‡ Bias (ESD): Mean bias and empirical standard deviation of the splitting values for the models that selected the correct covariate to split the node.

§ Number of models that selected the correct covariates at node h_1 and node h_3. Used as the denominator value when calculating the percentage of models that selected the correct covariate at node h_4.

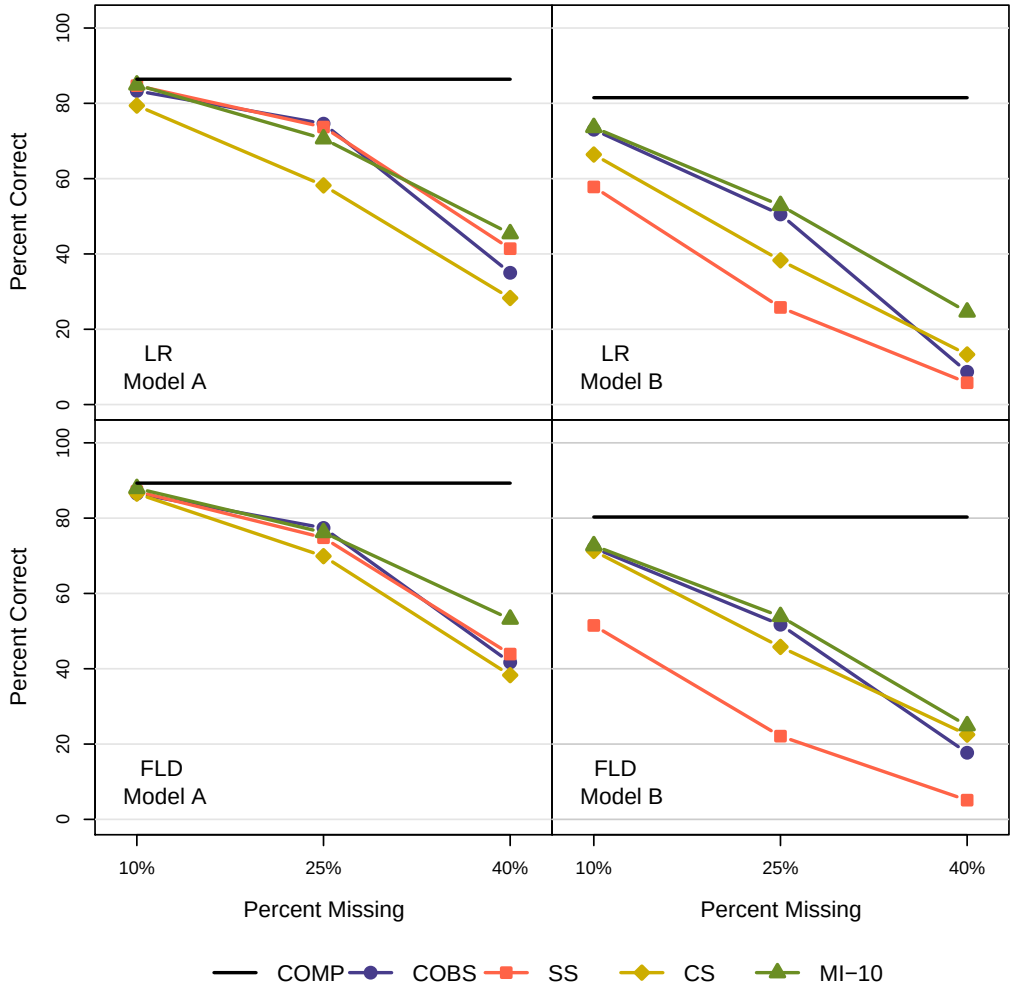

Figure 5.4: Percentage of completely correct models

5.3 MTLD-I APPLICATION

Our goal was to use the MTLD data to develop a prognostic model for identifying individuals who do not respond to treatment for depression. Specific covariates of interest, summaries of which are shown in Table 4.1, were depression, anxiety, cumulative illness, self esteem, age, duration of depression, and sleep quality. However, of the 151 individuals in our analysis sample, 37 (24.5%) did not have Pittsburgh Sleep Quality Index (PSQI) observations at the start of the acute phase. The PSQI is a self rated 19-item scale. Scores may range from 0 to 21, with higher values indicating greater and more severe sleep complications [39].

In total, two different tree models were created, each utilizing a different method to manage the missing PSQI observations. The first tree model employed Conversano and Siciliano's single imputation algorithm and the second model employed our proposed stochastic multiple imputation method with $M = 10$. Both tree models were grown with the log rank splitting statistic [23, 26]. To keep the models as parsimonious as possible, we required that at least 60 observations be present to split a node and that at least 20 observations be present in each terminal node. To be consistent with our simulation study, we did not utilize any pruning techniques.

The final trees and Kaplan Meier survival plots of the terminal nodes for the model utilizing Conversano and Siciliano's single imputation method are shown in Figure 5.5. The final trees and Kaplan Meier survival plots of the terminal nodes for our proposed multiple imputation algorithm are shown in Figure 5.6. Both models begin by splitting on anxiety, depression, and then sleep quality (PSQI). However, the CS model selected 14.5 as the best PSQI split and the MI-10 model selected 7.5 as the best PSQI split. After splitting on the PSQI score, the two models diverged further, one splitting on self esteem and the other on the PSQI once again.

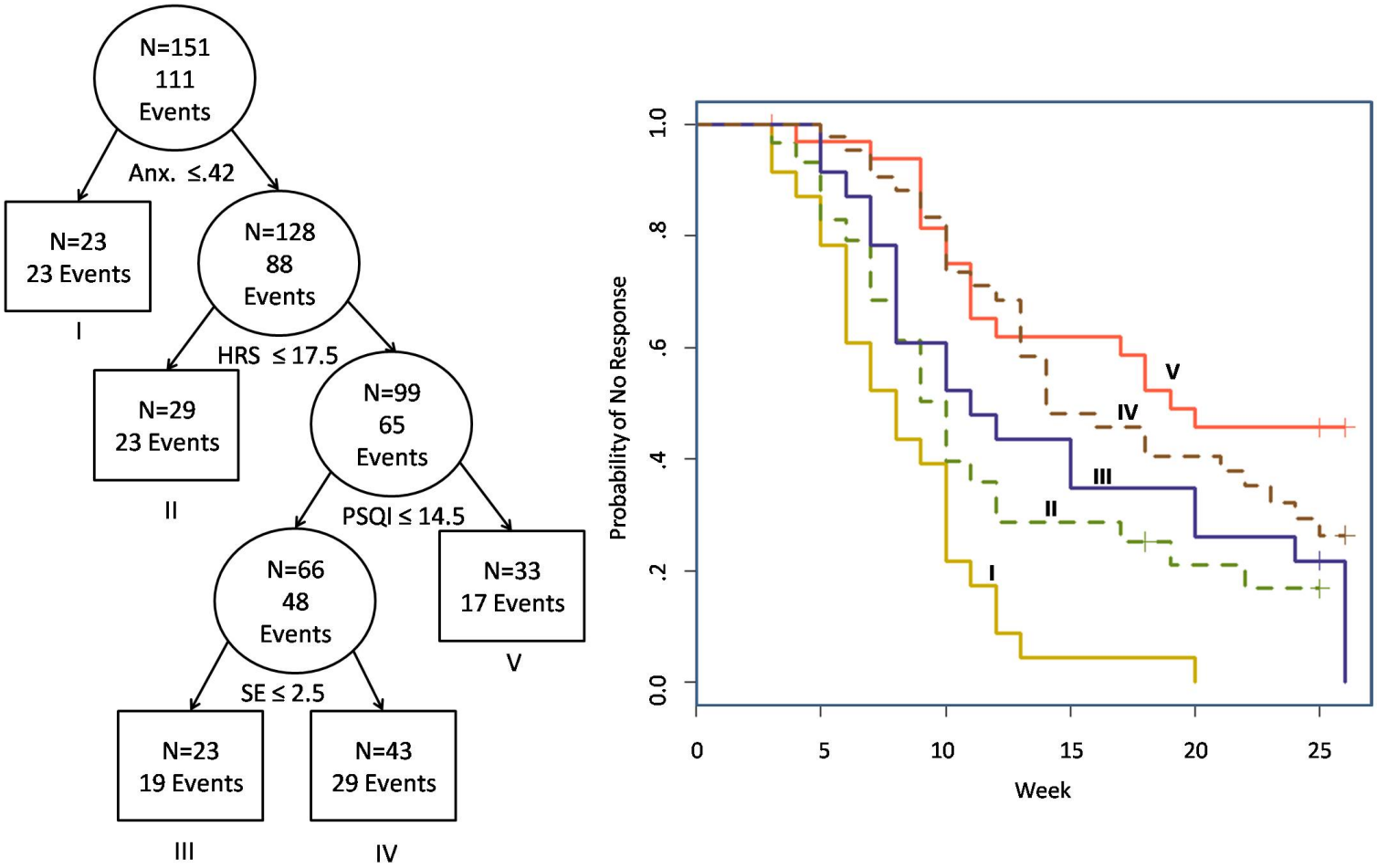

Figure 5.5: Conversano and Siciliano single imputation method (CS)

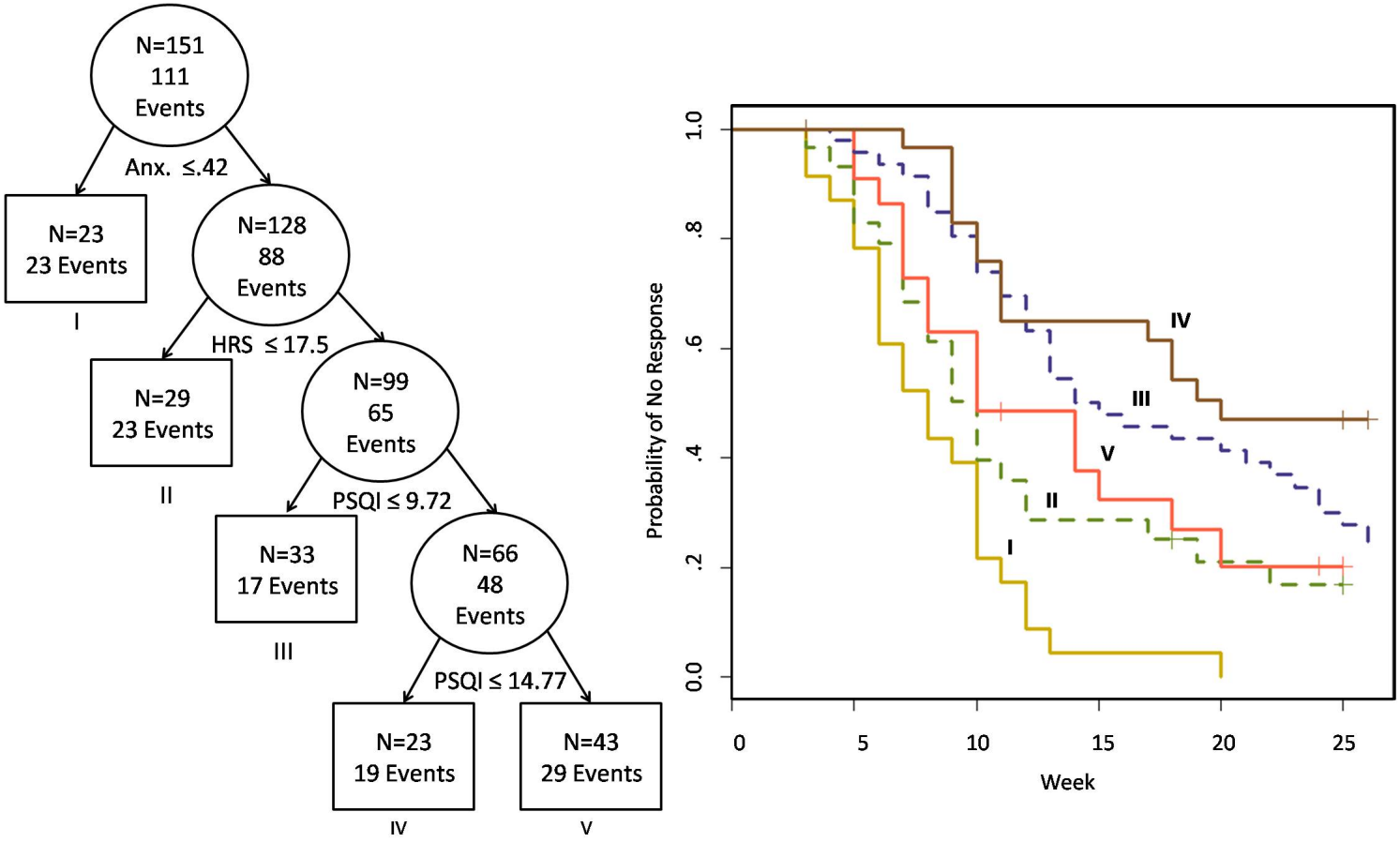

Figure 5.6: Multiple imputation method with M=10 (MI-10)

5.4 DISCUSSION

As shown in our application in Section 5.3, different methodologies for missing covariate data may easily result in different tree models. Consequently, we emphasize the importance of conscientiously selecting an appropriate method for accommodating missing data in tree-structured survival analyses. With respect to overall tree accuracy, our proposed method and the currently used methods were similar with only 10% missing data and with a simple covariate structure, regardless of the splitting statistic. With higher percentages of missing covariate data and complex tree structures, our tree-structured multiple imputation algorithm was more accurate. Overall, the full likelihood deviance was shown to outperform the log rank statistic, and the simpler covariate structure of model A resulted in more accurate results than the more complex covariate structure of model B.

Although results for our multiple imputation method are promising, it is important to note that there are many different data structures that could alter the accuracy of the methods, and they should be more closely examined before further conclusions are are made regarding a preferred method. Specifically, we believe that the accuracy of any method for managing missing covariate data in TSSA hinges upon the strength of the relationship between the covariate with missing data and the outcome, as indicated by its position in the tree. When a covariate with missing data has a relatively strong relationship with the outcome it may split near the top of the tree, and cases incorrectly classified may cause a cascading effect of inaccuracy. However, if the covariate with missing data has a relatively weak relationship with the outcome it may split near the bottom of the model, where there are fewer nodes below it that could be negatively affected by misclassified cases.

The accuracy of our method and other competing methods may also depend on the number of covariates with missing observations and to how many other covariates their missingness mechanisms are related. Simulation results showed that our proposed multiple imputation method had lower accuracy than other methods at nodes that split on the imputed covariate, but higher accuracy than other methods at the descendant nodes that split on the covariates associated with the missingness mechanism. The overall accuracy of our method could therefore change depending on which covariates are selected to split the nodes

in the tree, and may be further compounded by the positioning of these covariates within the tree, as discussed in the previous paragraph.

More research regarding the selection of the number of multiple imputations needs to be conducted. Generally, our simulation showed that the tree models with $M = 10$ imputations resulted in a higher accuracy of covariate selection and lower bias and ESD for the imputed covariate than the models with $M = 1$. However, because tree-structured analysis is already computationally intensive, it is important to determine the minimal number of imputations that can provide accurate and stable models. A related area of further research is the cutoff value utilized to select an imputed covariate and send its cases down the tree. We selected a cutoff value of $M/2$, but recognize that in other situations a different cutoff value may be more appropriate. For example, if a covariate with missing data is assumed to be vital to the tree structure, a cutoff value other than $M/2$ might be chosen to ensure that this covariate is selected.

In summary, it is important to carefully consider the hypothesized structure of the data and relative strengths of the relationships between covariates and the outcome when selecting a preferred method for handling missing-at-random covariate data. In our simulation study, when only 10% missing covariate data and a simple data structure were present, all of the methods we tested performed quite well compared to the models using the complete dataset. However, as both the missingness percentage and the complexity of the covariate structure increased, our method outperformed the other methods with respect to the overall accuracy of the survival tree.

6.0 USE OF RANDOM EFFECTS MODELS FOR INCORPORATING TEMPORAL FEATURES OF REPEATEDLY MEASURED COVARIATES

Tree-structured classification methods using only fixed-time predictor variables can result in accurate and practical prognostic models in a variety of different clinical settings. However, limiting the model to only baseline covariates may provide only partial information for the identification of patient risk groups due to the fact that changes in patient characteristics that occur after baseline ay be integral to understanding the disease process. We therefore propose to utilize temporal features of repeatedly measured covariates in order to glean additional information that may create more reliable prognostic groups.

As discussed by Laird and Ware [40], temporal features of repeatedly measured covariates can be derived with a two-stage random-effects model, typically described as either a growth model or a repeated-measures model. Growth curve models emphasize the natural developmental process to explain the within-subject variation for each individual. Alternatively, the classical repeated-measures model assumes that the individual effects remain constant over the time period of interest. In this paper, we discuss how the two-stage random-effects model can be used to extract individual features of repeatedly measured covariates and then how these features can be used as predictors in a tree-structured survival model. We also discuss how our proposed models can be used to classify new patients and predict their risk.

For late-life depression, we hypothesize that the linear rate of change of anxiety during treatment for depression is important for identifying risk groups that distinguish treatment resistant patients. To explore this hypothesis, we use a two-stage random effects model to estimate each individual's rate of change of anxiety, and use the set of these estimates in a tree-structured survival model to predict time to treatment response. We fit tree-structured survival models using traditional fixed-time covariates as well temporal features of covariates.

The empirical Brier score is used to assess and compare the predictive abilities of these models when used for classification and risk prediction.

6.1 TREE-STRUCTURED SURVIVAL MODELS INCORPORATING REPEATEDLY MEASURED COVARIATES

In this section, we begin by proposing new methodology to incorporate temporal features of repeatedly measured covariates into the traditional TSSA model. Unlike the traditional tree-structured survival model incorporating only fixed time covariates, however, our proposed methodology requires each new individual to have repeatedly measured clinical characteristics before they can be classified into a risk group. Therefore, in this section we also discuss how the resulting algorithm can be used for classification and risk prediction of new individuals based on repeatedly updated covariate information.

6.1.1 Incorporating Temporal Features Into the Tree Model

Each individual i, $i = 1, \ldots, N$, begins follow-up for the survival event of interest at a baseline time point, τ_i. Each individual is then followed until either his/her event time, T_i, or his/her time of censoring from the study, C_i, such that the observed outcome for each individual, i, is denoted by (T_i^*, δ_i), where $T_i^* = \min(Y_i, C_i)$, $\delta_i = 1$ when $T_i \leq C_i$ and $\delta_i = 0$ otherwise. The ordered, unique event times for all N individuals are denoted by t_l, $l = 1, \ldots, L$.

For each individual, i, we also observe a repeatedly measured covariate, W_i, on at least two discrete time points in the set $\mathcal{M} = \{t_j : j = 1, \ldots, J\}$. The subset of time points in $\mathcal{M}$ for which an individual, i, has observed measurements is denoted $\mathcal{M}_i = \{t_{i1}, \ldots, t_{iJ_i} \leq T_i^*\}$, where $\mathcal{M}_i \subseteq \mathcal{M}$ and J_i is the number of repeated measurements for the i^{th} subject. There is no requirement that the observed time points be equally spaced or correspond with the unique event times, t_l, $l = 1, \ldots, L$. The vector of corresponding observed repeatedly measured covariate values is denoted $\mathbf{W}_i = [W_i(t_{i1}), \ldots, W_i(t_{iJ_i})]'$.

We place no restrictions regarding the temporal relationship of the last observed time

point of the repeatedly measured covariate, t_{iJ_i}, and the onset of follow-up for the survival event, τ_i. However, we specifically consider two scenarios: (1) $t_{i1}, \ldots, t_{iJ_i} \leq \tau_i < T_i^*$ and (2) $\tau_i \leq t_{i1}, \ldots, t_{iJ_i} \leq T_i^*$. In scenario (1) we assume that all repeatedly measured covariate observations have been collected before the period of follow-up for the survival event of interest begins. Therefore, the number of repeatedly measured observations collected for each individual is not dependent on their outcome, T_i^*. In scenario (2), we assume that all repeatedly measured covariate observations occur during the period of follow-up for the survival event of interest. In this scenario, individuals with earlier outcomes may have fewer repeatedly measured covariate observations than individuals with later outcomes.

For either of the above scenarios, we propose to use features of repeatedly measured data as predictors in a tree-structured survival model through two steps:

1. Fit a two-stage random-effects model to estimate a selected feature of the repeatedly measured predictor for each individual; and
2. Include the individually estimated features in the predictor space for a TSSA model.

6.1.1.1 Step 1: Extracting a Selected Feature of a Repeatedly Measured Covariate

We propose to use a two-stage random-effects model to estimate a selected temporal feature of each individual's repeatedly measured covariate. This approach is ideal because it allows for each individual to be modeled separately while still borrowing information from those with more information, subsequently ensuring more accurate parameter estimates for individuals with less information. For scenario (2) in particular, this model characteristic helps to ensure that the extracted estimates for individuals with earlier outcomes (implying fewer repeatedly measured covariate observations) will not be artificially extreme and unstable compared to individuals with later outcomes (implying more repeatedly measured covariate observations). Furthermore, it does not require complete follow up or repeatedly measured covariate measurements at the same time for each individual [40, 41, 42].

We model the time dependent covariate for the i^{th} individual as

$$\mathbf{W}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\mathbf{b}_i + \boldsymbol{\epsilon}_i, \tag{6.1}$$

where $\boldsymbol{\beta}$ is a $p \times 1$ vector of unknown population parameters and $\mathbf{X}_i$ is a known $J_i \times p$ design matrix linking $\boldsymbol{\beta}$ to $\mathbf{W}_i$. Furthermore, $\mathbf{b}_i$ is a $k \times 1$ vector of unknown individual effects and $\mathbf{Z}_i$ is a known $J_i \times k$ design matrix linking $\mathbf{b}_i$ to $\mathbf{W}_i$ [40].

Following Laird and Ware [40], we assume that the individual parameter effect(s) are distributed as $\mathbf{b}_i \sim N(\mathbf{0}, \mathbf{G})$, where $\mathbf{G}$ is the between-subject variance/covariance matrix containing the set of between-subject variance/covariance parameters, denoted by $\boldsymbol{\gamma}_G$ when arranged as a vector. The vector of residual errors, $\boldsymbol{\epsilon}_i = [\epsilon_1, \ldots, \epsilon_{J_i}]'$, is assumed to be independent of $\mathbf{b}_i$ and distributed as $\boldsymbol{\epsilon}_i \sim N_{J_i}(\mathbf{0}, \mathbf{R}_i)$, where $\mathbf{R}_i$ is the $J_i \times J_i$ residual variance matrix containing the set of within-subject variance parameters, denoted by $\boldsymbol{\gamma}_R$ when arranged as a vector. The dimensions of $\boldsymbol{\gamma}_G$ and $\boldsymbol{\gamma}_R$ depend on constraints in the covariance structure. The repeatedly measured covariate, $\mathbf{W}_i$, is distributed as $\mathbf{W}_i \sim N_{J_i}(\mathbf{X}_i\boldsymbol{\beta}, \mathbf{V}_i)$, where $\mathbf{V}_i = \mathbf{Z}_i\mathbf{G}\mathbf{Z}_i' + \mathbf{R}_i$. Any missing observations on the repeatedly measured covariate are assumed to be either missing completely at random (MCAR) or missing at random (MAR) [3].

Given that all assumptions outlined above are met, the best linear unbiased estimates for the parameters in $\boldsymbol{\gamma}_G$ and $\boldsymbol{\gamma}_R$ can be obtained through an estimation method such as restricted maximum likelihood (REML), which reduces the likelihood to include only the variance parameters in $\boldsymbol{\gamma}_G$ and $\boldsymbol{\gamma}_R$. The resulting parameter estimates, $\hat{\boldsymbol{\gamma}}_G$ and $\hat{\boldsymbol{\gamma}}_R$, are inserted into the variance/covariance matrices $\mathbf{G}$ and $\mathbf{R}_i$ matrices to create $\hat{\mathbf{G}}$ and $\hat{\mathbf{R}}_i$, respectively. Subsequently, the overall variance estimate, $\hat{\mathbf{V}}_i = \mathbf{Z}_i\hat{\mathbf{G}}\mathbf{Z}_i' + \hat{\mathbf{R}}_i$, can be calculated.

To obtain estimates for $\boldsymbol{\beta}$ and $\mathbf{b} = [\mathbf{b}_1', \ldots, \mathbf{b}_N']'$, one standard method is to solve the mixed model equations given by Henderson [43]

$$\begin{bmatrix} \mathbf{X}'\hat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{X}'\hat{\mathbf{R}}^{-1}\mathbf{Z} \\ \mathbf{Z}'\hat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{Z}'\hat{\mathbf{R}}^{-1}\mathbf{Z} + \hat{\mathbf{G}}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\hat{\mathbf{R}}^{-1}\mathbf{W} \\ \mathbf{Z}'\hat{\mathbf{R}}^{-1}\mathbf{W} \end{bmatrix}, \tag{6.2}$$

where $\mathbf{X} = [\mathbf{X}_1', \ldots, \mathbf{X}_N']'$, $\mathbf{Z} = [\mathbf{Z}_1', \ldots, \mathbf{Z}_N']'$, $\mathbf{W} = [\mathbf{W}_1', \ldots, \mathbf{W}_N']'$ and $\hat{\mathbf{R}}$ is a block diagonal of $\hat{\mathbf{R}}_i$, $i = 1, \ldots, N$. Assuming $\hat{\mathbf{G}}$ is nonsingular, the solutions to these equations are written as

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{X})^{-}\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{W} \tag{6.3}$$

$$\hat{\mathbf{b}} = \hat{\mathbf{G}}\mathbf{Z}'\hat{\mathbf{V}}^{-1}(\mathbf{W} - \mathbf{X}\hat{\boldsymbol{\beta}}) \tag{6.4}$$

where $\hat{\mathbf{V}}$ is a block diagonal of $\hat{\mathbf{V}}_i$, $i = 1, \ldots, N$.

After obtaining parameter estimates $\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{b}}$, we can estimate of the feature of interest for an individual, i, based on the estimated population and individual-specific effects of the temporal feature,

$$\hat{B}_i^* = \hat{\beta}^* + \hat{b}_i^*,$$

where $\hat{\beta}^* \in \{\hat{\boldsymbol{\beta}}\}$ and $\hat{b}_i^* \in \{\hat{\mathbf{b}}_i\}$. Thus, the estimated temporal feature for the i^{th} individual, $\hat{B}_i^*$, is composed of the population estimate, $\hat{\beta}^*$, and the additional deviation of the i^{th} individual from the population estimate, b_i^*. The vector containing the estimated temporal features for the N individuals is denoted as $\hat{\mathbf{B}}^* = [\hat{B}_1^*, \ldots, \hat{B}_N^*]'$.

6.1.1.2 Step 2: Creating the Tree Model

The set of individual temporal feature estimates, $\hat{\mathbf{B}}^*$, and any baseline covariates of interest are included in the predictor space used to create the full tree model, H_{MAX}. If a pruning algorithm is used to select a more parsimonious tree model, we assume that the entire set of estimates, $\hat{\mathbf{B}}^*$, is the fixed sample, and any test or cross-validation samples are randomly drawn from this fixed sample. The final, pruned tree is denoted by H^*.

6.1.2 Classification

An advantage of the resulting tree structured algorithm, H^*, is that it can be used to classify new individuals (not in the original cohort) into risk groups. When the temporal feature, $\mathbf{B}^*$, is selected to split a node in H^*, the classification of a new individual, k, can only be accomplished after obtaining an estimate of the new individual's temporal feature, $B_k^* = \beta^* + b_k^*$, where

$$\mathbf{b}_k \;=\; \mathbf{G}\mathbf{Z}_k^{'}\mathbf{V}_k^{-1}(\mathbf{W}_k - \mathbf{X}\boldsymbol{\beta}), \tag{6.5}$$

with $\beta^* \in \boldsymbol{\beta}$ and $b_k^* \in \{\mathbf{b}_k\}$. Thus, we must first observe the new individual's repeatedly measured covariate of interest on at least two time points and obtain estimates of both the population and individual components of their temporal feature.

One possible method for obtaining estimates of β^* and b_k^* is to fit a new random-effects model that incorporates the data from individual k into the original dataset used to create H^*. However, for investigators who have interest in predictive classification, such an approach would not be feasible unless they had access to the data fitted to the original N individuals. To estimate the population and individual components of the feature of interest without access to the original database, we therefore make the assumption that the population estimates, as well as the variance and covariance estimates, would not change significantly if the new individual, k, had been included in the original dataset.

Under this assumption, the population components, $\boldsymbol{\beta}$, can be set equal to the corresponding population estimates, $\hat{\boldsymbol{\beta}}$, obtained in the original random-effects model. However, the individual components, $\mathbf{b}_k$, need to be estimated based on the specific information gathered from the new individual, k. To estimate the random components, we must therefore first obtain measurements of the repeatedly measured covariate, $\mathbf{W}_k$, on multiple discrete ordered times t_{kj}, $j = 1, \ldots, J_k$. The set of these discrete time points for an individual, k, is denoted $\mathcal{M}_k$, with $\mathcal{M}_k \subseteq \mathcal{M}$. The corresponding observed covariate values are denoted $\mathbf{W}_k = [W_k(t_{k1}), \ldots, W_k(t_{kJ_k})]'$. We insert these repeated measurements into the equation for the empirical best linear unbiased predictor specific to individual k (6.5), where the estimates of the parameters in $\boldsymbol{\gamma}_R$ and $\boldsymbol{\gamma}_G$ are assumed to be the same as in the original two-stage random effects model.

The random component(s) of the feature for the new individual, k, are then estimated as

$$\tilde{\mathbf{b}}_k \;=\; \hat{\mathbf{G}}\mathbf{Z}_k'\hat{\mathbf{V}}_k^{-1}(\mathbf{W}_k - \mathbf{X}_k\hat{\boldsymbol{\beta}}), \tag{6.6}$$

where $\hat{\mathbf{V}}_k = \mathbf{Z}_k\hat{\mathbf{G}}\mathbf{Z}_k' + \hat{\mathbf{R}}_k$. In this equation, $\hat{\mathbf{G}}$ and $\hat{\boldsymbol{\beta}}$ have the same dimensions and components as in original two-stage random effects model because they do not depend on the set of time points on which the new individual has observed data, $\mathcal{M}_k$. The matrices $\mathbf{Z}_k$, $\mathbf{X}_k$, and $\hat{\mathbf{R}}_k$, however, do depend on the specific time points in $\mathcal{M}_k$ and therefore may need to be recreated for new individuals being classified. The random- and fixed-effects design matrices, $\mathbf{Z}_k$ and $\mathbf{X}_k$, are $J_k \times p$ and $J_k \times r$, respectively, with rows corresponding to the

specific time points in $\mathcal{M}_k$. The within-subject error matrix, $\hat{\mathbf{R}}_k$, has dimensions $J_k \times J_k$, with both rows and columns corresponding to the specific time points in $\mathcal{M}_k$.

We extract the random component of the feature of interest, $\tilde{b}_k^*$ from the vector of newly calculated random effect estimates, $\tilde{\mathbf{b}}_k$, and also the population component of the feature of interest, $\hat{\beta}^*$ from the vector of fixed effect estimates from the original model, $\hat{\boldsymbol{\beta}}$. The feature of interest for the new individual k is then calculated as $\tilde{B}_k^* = \hat{\beta}^* + \tilde{b}_k^*$. This estimate, along with other observed baseline values, is used to classify individual k into a terminal node $h \in \tilde{H}^*$.

6.1.3 Risk Prediction

6.1.3.1 Scenario (1) In scenario (1), we assume the values of the repeatedly measured covariate have already been observed before the onset of follow-up for the survival event, τ_k. Therefore, the classification of a new individual, k, can occur immediately at time τ_k, as long as baseline covariate values have been observed. The Kaplan Meier survival estimate [12], commonly used to summarize survival in the terminal nodes of tree-structured survival models, can thus be used without modification to summarize each terminal node $h \in \tilde{H}^*$. The Kaplan Meier survival estimate for a terminal node, h, at a time, t, is defined as

$$\hat{S}_h(t) = \begin{cases} 1 & \text{if } t < t_1 \\ \prod_{t_l \leq t}[1 - \frac{d_{lh}}{R_{lh}}] & \text{if } t_1 \leq t \end{cases}, \tag{6.7}$$

where d_{lh} is the number of events in terminal node h at the event time, t_l, R_{lh} is the number of individuals in node h at risk just prior to t_l, and d_{lh}/R_{lh} is the estimate of the conditional probability that an individual in node h who survives just prior to t_l experiences the event at t_l. Using this survival estimate we can also obtain percentiles of the survival estimate $\hat{S}_h(t_{(p)}) \leq 1 - p$, e.g., selecting $p = .5$ estimates the median survival time.

6.1.3.2 Scenario (2) In scenario (2), the repeatedly measured covariate is observed during the follow-up for the survival event. Because the temporal feature requires observing an individual, k, on multiple time points, it is necessary to wait until a time, t_{kJ_k}, where $t_{kJ_k} > \tau_k$ and $J_k \geq 2$, to classify the new individual. In the original sample utilized to create

H^*, there may have been event times, t_l such that $t_l \leq t_{kJ_k}$ for some value of l. However, because individual k has not yet had an event at this time, we must estimate survival conditional on no event occurring prior to the time of classification, t_{kJ_k}. For individuals in node h, the conditional probability of survival beyond time t, given survival to time t_{kJ_k}, is defined as

$$\hat{S}_{t_{kJ_k}h}(t) = \begin{cases} 1 & \text{if } t < t_{kJ_k} \\ \prod_{t_{kJ_k} \leq t_l \leq t}[1 - \frac{d_{lh}}{R_{lh}}] & \text{if } t \geq t_{kJ_k} \end{cases}, \tag{6.8}$$

where d_{lh} and R_{lh} are defined as in subsection 6.1.3.1. We can also obtain percentiles for the residual life given survival until time t_{kJ_k}, defined as $\hat{S}_{t_{kJ_k}h}(t_{(p)}) \leq 1 - p$. Selecting $p = .5$ results in the median survival given no event has occurred by the time of classification, t_{kJ_k}. Inference for this estimate, including confidence intervals and tests for differences between nodes at later time points, is discussed by Jeong *et al.* [44].

6.2 MTLD-I APPLICATION

Our goal was to create an algorithm that could be used throughout the course of acute treatment to help clinicians identify patient risk and subsequently assist them in making treatment decisions. Therefore, we only used data from the onset of the acute treatment phase until the time of treatment response or withdrawal from the acute phase. Our outcome of interest was time until response to depression treatment, defined as an HRSD score ≤ 10 for at least three consecutive weeks. Covariates of interest, summarized in Table 4.1, include the Brief Symptom Inventory (BSI) anxiety subscale, the interpersonal support evaluation list score (ISEL) self-esteem subscale, the Cumulative Illness Rating Scale for Geriatrics (CIRS-G), age at the onset of acute treatment, gender, the number of years since their first depressive episode (duration), and also the HDRS. We analyzed a subset of N=151 individuals with baseline covariates observed at the onset of acute treatment and at least three consecutive HRSD observations during the acute treatment phase.

6.2.1 Implementation of Traditional TSSA Methodology

We created risk groups with LeBlanc and Crowley's [21] full likelihood deviance survival tree methodology, using only baseline covariates in the predictor space. To keep the resulting tree algorithm at a clinically meaningful size, we incorporated a complexity parameter of $\alpha = .02$, required at least 60 observations to split a node and required at least 20 observations in each descendant node. Ten-fold cross-validation was used to prune the tree. The *rpart* package [4, 29] within the statistical program, *R* [30], was utilized to generate the model.

As shown in Figure 6.1, baseline anxiety is identified as the most important predictor of treatment response. For those individuals with higher baseline anxiety, baseline depression further delineates the risk groups. The group at the highest risk for not responding to treatment has a median of 19 weeks until response. Thirty-three out of 57 individuals in this terminal node responded to treatment by the end of the 26 week acute treatment phase.

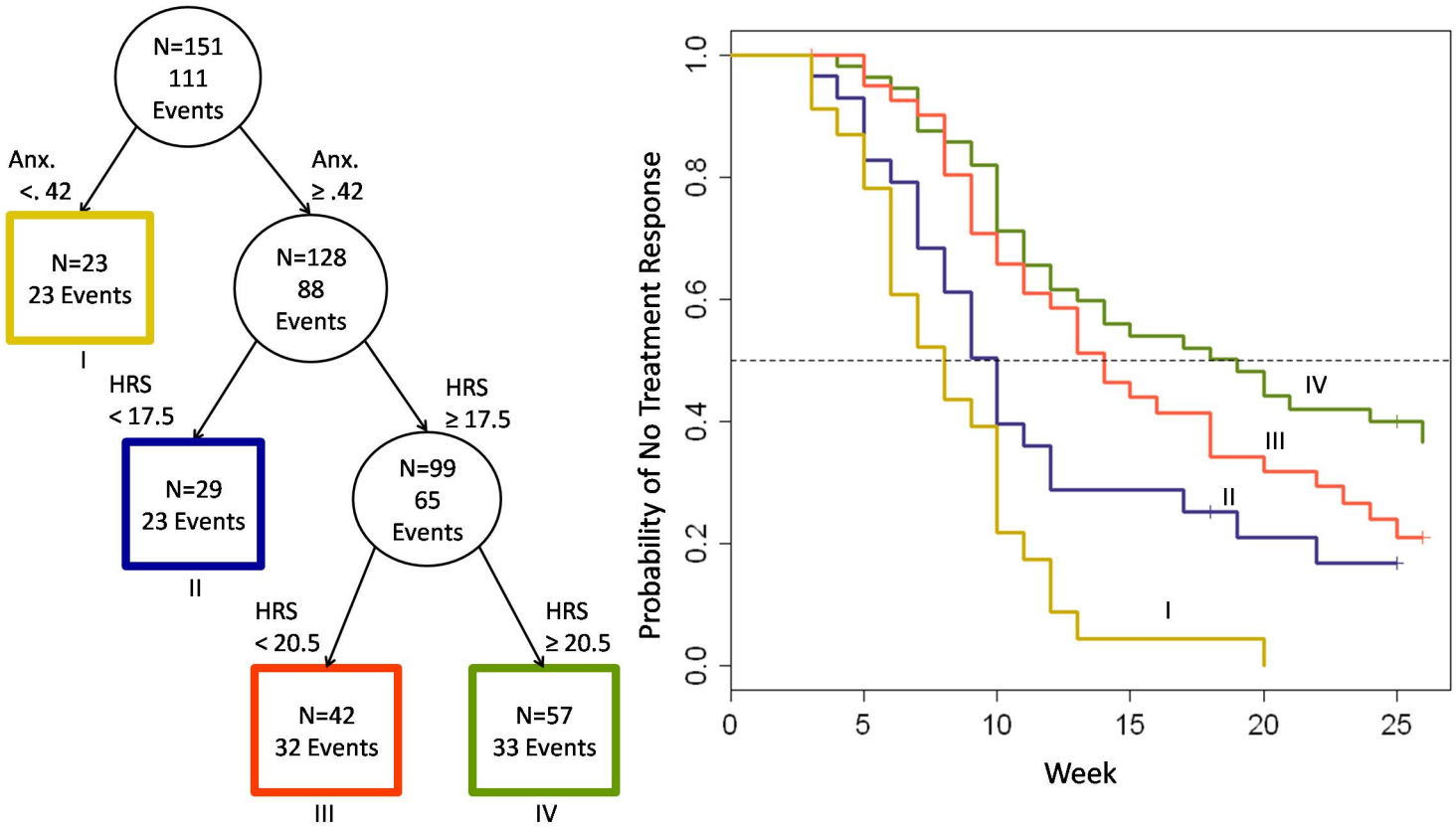

Figure 6.1: Model using only baseline covariates

6.2.2 Implementation of Methodology for Incorporating Temporal Features in TSSA

We hypothesized that individuals whose anxiety decreased more slowly during acute treatment would be less likely to have their depressive symptoms subside by the end of the 26 week acute treatment phase. To explore this hypothesis, we first fit a two-stage random effects model to estimate the rate of change of anxiety for each individual. We used the "MIXED" procedure in SAS version 9 [45], with code as follows:

```
PROC MIXED DATA=anxlong COVTEST;
   CLASS subject;
   MODEL anxiety=week;
   RANDOM intercept week / SUBJECT=subject TYPE=UN G V=i;
   ODS OUTPUT SOLUTIONR=features;
RUN;
```

In the first line of our code, "`anxlong`" is the name of the dataset holding the observations and "`COVTEST`" requests a likelihood ratio test to determine whether each of the random effect parameters in $\mathbf{\Gamma}_G$ are significantly different from 0. In the "`CLASS`" statement, "`subject`" is the unique identification of each cluster of repeatedly measured observations. In the "`MODEL`" statement, "`anxiety`" is the observed values of the repeatedly measured covariate, and "`week`" is the quantification of when each of the repeated measurements in "`anxiety`" was observed.

In the `RANDOM` statement, we selected the week and intercept as the effects to be modeled individually. The option `TYPE=UN` in this statement creates an unstructured $\mathbf{G}$ matrix containing between-subject variance/covariance parameters in the vector $\mathbf{\Gamma}_G$. Because we excluded a `REPEATED` statement, the within-subject variance matrix is assumed to be $\mathbf{R} = \sigma_e^2\mathbf{I}$. To assist in the classification of new individuals, we requested to have SAS output the estimated between-subject variance/covariance matrix, $\hat{\mathbf{G}}$, as well as the estimated overall variance/covariance matrix for an individual i, $\hat{\mathbf{V}}_i$. The `ODS` statement requests that the set of predicted random intercept and slope effects, $\hat{\mathbf{b}}$, be exported to a new dataset, "`features`".

The between-subject variance-covariance matrix ($\mathbf{G}$), the residual error (σ_e^2), and the fixed effects parameters ($\boldsymbol{\beta} = [\beta_0, \beta_1]'$) were estimated as

$$\hat{\mathbf{G}} = \begin{bmatrix} .6575 & -.03882 \\ -.03882 & .004470 \end{bmatrix},$$

$\hat{\sigma}_e^2 = .1451$, and $\hat{\boldsymbol{\beta}} = [1.2383, -.07239]'$, respectively. Each individual-specific rate of change estimate, $\hat{b}_{i1}$, was added to the population estimate, $\hat{\beta}_1$, to obtain each individual's estimated rate of change of anxiety per week.

To create the tree model illustrating proposed methodology, we used the same baseline covariates as in the traditional model as well as our estimated rate of change covariate. We used the same methods for constructing the new tree model as were used for constructing the traditional model. The resulting algorithm is shown in Figure 6.2.2. As in the traditional model, baseline anxiety is the first predictor to split the sample. However, for individuals with higher baseline anxiety, the rate of change of anxiety is the next most important predictor of time to treatment response. Both baseline anxiety and rate of change of anxiety are then split upon once again.

Of particular interest in this model is the group of 32 individuals in terminal node V, of which only eight responded to treatment. Individuals classified into this terminal node begin acute treatment with "high" BSI anxiety ($> .482$) that subsequently changes at a rate greater than -.1 units per week. Thus, an individual with a "high" baseline BSI anxiety level that either very slowly decreases, remains constant, or increases over time is at risk of not responding to treatment.

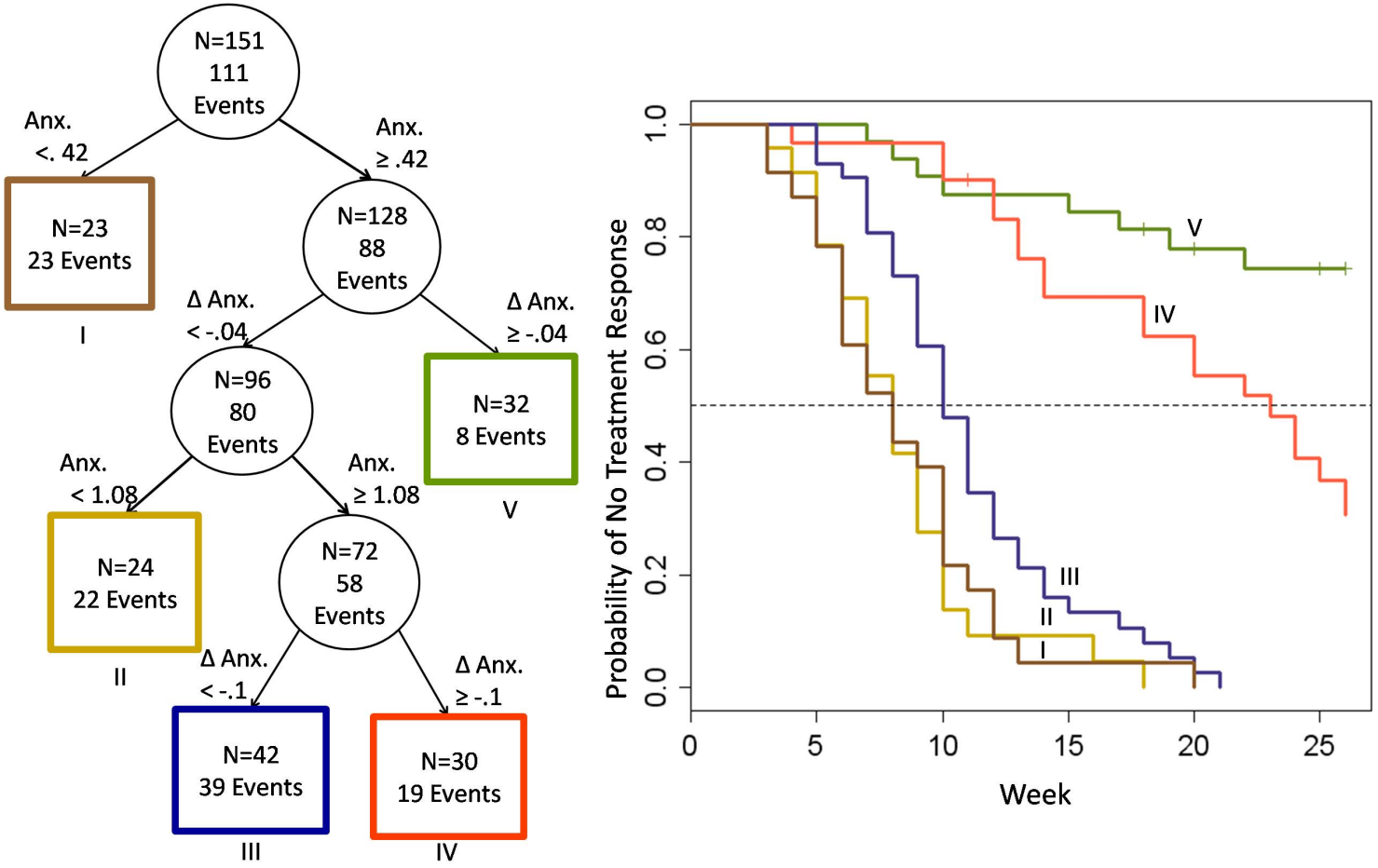

Figure 6.2: Model using baseline covariates plus the rate of change of anxiety during acute treatment (Δ Anx.)

6.2.3 Assessing Predictive Abilities of Tree Models

The Kaplan Meier plots of the terminal nodes of our proposed model, shown in Figure 6.2.2, visually illustrate how the classification groups created by our proposed methodology can more clearly discriminate the at-risk individuals than the classification groups created with the traditional methodology, shown in Figure 6.1. However, we also desired to quantitatively asses the predictive abilities of our models when used throughout the course of the 26-week treatment to classify individuals into risk groups.

The empirical Brier score [8] is a useful statistic for our purposes because it allows us to compare the predictive accuracy of the model when used throughout treatment, while also accounting for dropouts due to censored observations. This statistic is based on the difference between the indicator variable for the event outcome, δ_i, and the estimated probability of the event at time t_l. The hypothetical case of perfect foresight results in a Brier score equal to zero, that is, no difference between prediction and true failure status. A Brier score of .25 corresponds to the trivial rule of always assigning 50% probability of survival, and therefore is the upper boundary of a reasonable prediction. The maximum Brier score, indicating perfect "inverse" foresight, takes the value of one [8].

When repeatedly measured covariates are employed, we incorporate modifications presented by Schoop *et al.* [38] to estimate the Brier score. At each event time, t_l, we can estimate the empirical Brier score for our data using

$$BS(t_l) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1(Y_i^* \leq t_l)\hat{S}_h(t_l; W_i^{t_l})\delta_i}{\hat{G}(Y_i^*)} + \frac{1(Y_i^* > t_l)\{1-\hat{S}_h(t_l; W_i^{t_l})\}}{\hat{G}(t_l)}\right], \quad (6.9)$$

where $\hat{S}_h(t_l; W_i^{t_l})$ is the Kaplan Meier estimate of the survival distribution in terminal node h at event time t_l. Classification of an individual, i, into a terminal node, h, is determined by his/her covariate history up to time t_l, that is, $W_i^{t_l} = \{W_i(t_{ij}), t_{i1} \leq t_{ij} \leq t_l\}$. The indicator function $1(Y_i^* \leq t_l)$ takes the value of one for individuals who had either an event or censoring before the event time t_l and zero otherwise. Likewise, the indicator function $1(Y_i^* > t_l)$ takes the value of one for individuals who have not yet had a censoring or an event by the event time t_l and zero otherwise. The overall Kaplan Meier estimate of the censoring distribution is denoted by $\hat{G}$, which is not node-specific due to the assumption that

the censoring and survival distributions are independent. In addition to simply assessing the predictive ability of a model at one event time, t_l, the empirical Brier score can also be used to quantify a model's predictive abilities over all L event times, as follows

$$IBS = \frac{1}{L}\sum_{l=1}^{L} BS(t_l). \tag{6.10}$$

Given our relatively small sample size of N=151, we used a jackknife approach to assess the predictive abilities of both the traditional and proposed models. For the traditional model, we created subsets $\mathcal{L}^{-i}$, $i = 1, \ldots, N$, by iteratively removing the i^{th} individual from the full sample set, $\mathcal{L}$. Using individuals in a subset $\mathcal{L}^{-i}$, we grew and pruned a survival tree, H^{-i} and then classified the excluded observation, i, into a terminal node $h \in H^{-i}$ using only baseline covariate values, *i.e.*, $W_i^{t_l} = W_i(t_{i1})$. We then calculated the empirical Brier score contribution for individual i at each event time t_l. This process was repeated for each subset $\mathcal{L}^{-i}$ and excluded individual i, $i = 1, \ldots, N$. The resulting contributions were averaged together at each time point, t_l, $l = 1, \ldots, L$, to obtain each Brier score estimate ($BS(t_l)$). We also averaged the Brier scores over all time points to obtain the integrated Brier score estimate (IBS).

A similar approach was used to calculate the empirical Brier scores for the model based on our proposed methodology. As before, we first created subsets $\mathcal{L}^{-i}$, $i = 1, \ldots, N$. For individuals in a subset $\mathcal{L}^{-i}$, we fit a random effects model and included the resulting set of estimated linear rates of change into the predictor space to create the tree model H^{-i}. To calculate the Brier score contribution at an event time, t_l, for the excluded individual, i, we used his/her covariate history, $W_i^{t_l}$, as well as the fixed parameter estimates $\hat{\beta}_1$, $\hat{\sigma}_e$, and $\hat{\mathbf{\Gamma}}$ from the random coefficient model created with the sample $\mathcal{L}^{-i}$, to estimate his/her rate of change. Using this value and corresponding baseline covariate values, we classified the i^{th} individual into a terminal node $h \in \tilde{H}^{-i}$ and subsequently estimated their Brier score contribution. We repeated this process for event times t_l, each time allowing only the covariate history $W_i^{t_l}$ to be utilized to estimate the rate of change of the individual being classified. Consequently, the predicted rate of change of individual i, as well as their predicted terminal node, could be different depending on the event time, t_l, at which their Brier score contribution was calculated. At each event time, t_l, the individual contributions were averaged together to

obtain $BS(t_l)$. The integrated Brier score (IBS) was also calculated by averaging over the Brier scores at all time points.

Figure 6.3 displays the relative predictive accuracies of the two models when used to classify individuals at each of the event times t_l, $l = 1, \ldots, L$. Through week six, the two models have extremely similar levels of accuracy, and between weeks six and 12 the baseline model is shown to be slightly more accurate. After week 12, our proposed method shows a drastic improvement over the model created using only fixed-time covariates. One possible explanation for this improvement is that a larger number of repeatedly measured observations are being used to estimate each individual's rate of change, resulting in more stable feature estimates and consequently more accurate terminal node classification. Overall, the trees using our proposed methodology have an integrated Brier score of .1594, which is better than .1823 from the traditional model. Both are an improvement over a trivial prediction rule, which would result in an integrated Brier score of .25.

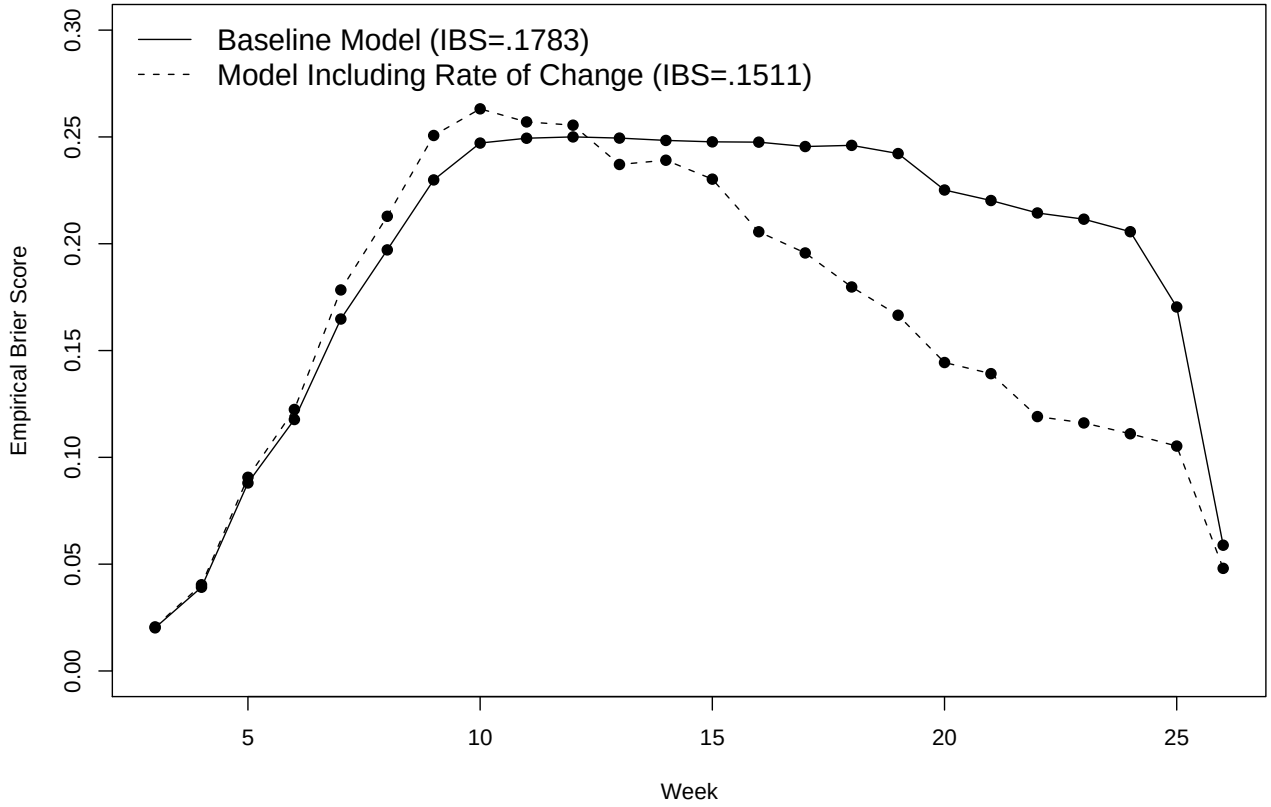

Figure 6.3: Empirical Brier Score at each event time.

6.3 DISCUSSION

Temporal features of repeatedly measured covariates can be important indicators of an upcoming event, and including them in the predictor space for a TSSA model may help clinicians identify reliable prognostic classification groups. For example, by applying our methodology to the MTLD data, we found that an individual with "high" baseline anxiety that does not decrease quickly enough over time is at the highest risk of not responding to treatment for depression. By using such an algorithm to identify such at-risk individuals during acute treatment, clinicians can adapt their treatment strategies to more appropriately fit the needs of their patients.

Our proposed approach has many advantages that lead to an accurate, reliable, and flexible model. By using a two-stage random-effects model to estimate temporal features, we can allow missing and unequally spaced repeatedly measured covariate data. Furthermore, using a temporal feature as a predictor variable in a tree models provides the flexibility to employ any desired splitting statistic. Of course, there are assumptions with our methodology. First, when fitting the random-effects model, we make the assumption that individuals with early events have either missing completely at random (MCAR) or missing at random (MAR) repeatedly measured covariate data. Typically, the MAR assumption is more reasonable, and implies that the missing covariate data are not missing because of their own unobserved values, but because of the value of another covariate. If neither of these assumptions are realistic for the dataset being used, it would be possible to use a pattern mixture model to account for covariate data that is not missing not at random (NMAR). Another assumption made in our model is that the trajectory of each individual's time-dependent covariate can be modeled by a polynomial function. With respect to our illustration, we specifically assume that anxiety changes linearly throughout the entire course of acute treatment and up to the observed outcome, Y_i^*. If anxiety were only assumed to change linearly only up to a certain time point t_j^*, one could alternatively model the rate of change of anxiety only during weeks t_j, $t_1 \leq t_j \leq t_j^*$.

One of the major advantages of a tree-structured model lies in its simplistic interpretation, providing a convenient algorithm for use in a clinical setting. However, when using

our proposed model the estimation of a new individual's temporal feature can not simply be done "in one's head". A computer program allowing the clinician to estimate the temporal feature based on updated covariate information would be required. Also, when repeatedly measured covariates are collected during the follow-up for survival, as in scenario (2), a new case can only be classified after multiple repeatedly measured covariate observations have been observed. When the treatment decision based on the resulting classification is time-sensitive, it may be of interest to estimate the feature early in the follow-up process. In this situation, the goal would be to collect the necessary observations as early as possible, while still estimating the feature with enough accuracy to place the individual into the correct prognostic group. We view this challenge as a trade-off for the additional prognostic information gained from the model utilizing a selected temporal feature of a repeatedly measured covariate.

7.0 FUNCTIONALS IN TIME-DEPENDENT TSSA

Bacchetti and Segal [9] and Huang, Chen et al. [10] propose methods to incorporate repeatedly measured covariates into TSSA. Just as in fixed time TSSA, the goal of time-dependent TSSA methodology is to detect the covariate value that divides the data into two groups with more homogeneous risk than the initial sample. However, in time-dependent TSSA, we test all covariate values, regardless of the time at which they are observed. After the tree model is created it is used for classification in the same manner as a TSSA model with only fixed covariates.

When time-dependent covariates are used, it is important to make a decision regarding the type of time-dependence present in the data [6]. For example, Gail [7] proposed that functionals of repeatedly measured serial cancer marker values be used to model the risk of recurrent disease in the Cox model. Examples provided by Gail include the marker value itself, the marker value at a previous time point, an indicator of whether the marker has increased over a prespecified level, and the rate of change of the marker from one time point to another. We believe that such repeatedly measured functionals could also be useful for creating prognostic groups, and therefore propose to extend Gail's work to be used with time-dependent TSSA methodology proposed by Bacchetti and Segal [9].

When utilizing functionals of continuous covariates in time-dependent TSSA, it is only realistic to assume that these values are not monotonically increasing or decreasing with time. Although the proposed time-dependent TSSA methodology technically accounts for this situation, the vast majority of work by Bacchetti and Segal [9] and Huang, Chen et al. [10] has been focused on testing and applying the much simpler scenario where repeatedly measured covariates are changing monotonically over time. A case in point is the simulation study by Huang et al., which only utilized monotonically changing repeatedly measured

covariates, as well as the applications by both Bacchetti et al. and Huang et al., which used calendar time and a transplant status indicator variable, respectively, as repeatedly measured covariates.

In this chapter, we propose methodology for time-dependent TSSA that clarifies issues regarding the use of non-monotonically increasing and decreasing repeatedly measured covariates, as well as present various functionals that are appropriate for use in TSSA. We then present simulation studies that assess the accuracy of time-dependent TSSA methodology with continuous covariates that are non-monotonically increasing over time. Our methodology is then applied to the MTLD-data, where we create and compare models using various functionals of repeatedly measured anxiety. The empirical Brier score is used to quantify the predictive abilities of these models throughout the course of acute treatment.

7.1 METHODOLOGY

7.1.1 Functionals of Time-Dependent Covariates

The observed outcome for each individual is denoted $T_i^* = \min(T_i, C_i)$, where $\delta_i = 1$ if $T_i^* = T_i$ and 0 otherwise. The set of J unique discrete event times for all individuals is denoted $\mathcal{M} = \{t_j : j = 1, \ldots, J\}$, such that each $T_i^* \in \mathcal{M}$. Each individual also has a set of repeatedly measured covariates $\mathbf{W}_i(t_{ij}) = [w_i(t_{i1}), \ldots, w_i(t_{iJ_i})]'$, not necessarily corresponding to the event times in $\mathcal{M}$.

In order for an event time, t_j, to be informative for time-dependent TSSA, it is necessary to have at least two individuals in the risk set, $\mathcal{R}_j$, the set of R_j individuals at risk just prior to time t_j. Furthermore, we also require that at least two individuals in $\mathcal{R}_j$ have an observed repeatedly measured covariate value at time t_j so that they can be assigned to a daughter node and the two-sample rank statistic contribution for time t_j can be calculated. One complication with this requirement is that sometimes individuals do not have an observed repeatedly measured covariate value at (or proximate to) a given event time t_j, even if they are still in the risk set, $\mathcal{R}_j$. To remedy this problem, we create a repeatedly

measured covariate, $\gamma_i(t_j)$, out of the time-dependent covariate $W_i(t_{ij})$ with observed values corresponding to the set of event times t_j for which individual $i \in \mathcal{R}_j$. The set of these time points for individual i is denoted $\mathcal{M}_i$, with $\mathcal{M}_i \subseteq \mathcal{M}$. For each individual, i, we also define a repeatedly measured indicator variable, $\delta_i(t_j)$, taking values at times t_j in the set $\mathcal{M}_i$, such that $\delta_i(t_j) = 1$ when $T_i^* = t_j = T_i$ and zero otherwise.

Gail provides example functionals that can delineate the relationship between the repeatedly measured covariate and the outcome, when used in the Cox model. However, when functionals are used in TSSA, the focus is on detecting a splitting value based on that functional. Examples of functionals, proposed by Gail [7], that are relevant for tree structured survival analysis include

$$\begin{aligned}
\mathbf{Z}_1(t_j) &= \boldsymbol{\gamma}(t_j), \\
\mathbf{Z}_2(t_j) &= \boldsymbol{\gamma}(t_{j-1}), \\
\mathbf{Z}_3(t_j) &= \boldsymbol{\gamma}(t_j) - \gamma(t_1), \quad \text{and} \\
\mathbf{Z}_4(t_j) &= \{\boldsymbol{\gamma}(t_j) - \boldsymbol{\gamma}(t_j - \Delta)\}/\Delta.
\end{aligned}$$

The first functional, $\mathbf{Z}_1(t_j)$, is simply the observed covariate measurement at each time, and would be used if it was believed that the updated covariate measurement could provide a more beneficial splitting value than just the measurements at one time point. The second functional, $\mathbf{Z}_2(t_j)$, is a lag functional which could be used to asses whether the predictor space should be split based on a previously defined value of a covariate. If this functional were incorporated in a tree model, the covariate value observed at time t_{j-1} would be used to classify an individual at a time t_j, instead of the observed value at time t_j. The third functional, $\mathbf{Z}_3(t_j)$, is a change from baseline functional, used when the magnitude of change from baseline, as opposed to the specific covariate value, is more important for classifying individuals. And finally, the functional $\mathbf{Z}_4(t_j)$ could be used to assess whether rate of change is important in classifying individuals into prognostic groups. Using this functional, a rate-of-change cut-off value would be defined, such that individuals with covariate values changing below or above a certain level would be at higher risk.

7.1.2 Growing the Time-Dependent Tree Model

The tree-growing process, when incorporating repeatedly measured covariates, is the same as with fixed time covariates with respect to the fact that for each node h it tests each possible split, s_h, selects the best split, s_h^*, based on a splitting criterion, $G(s_h)$, and continues splitting on each subsequent daughter node as long as there are a minimum observations. However, in time-dependent TSSA, a splitting criterion is used to quantify the value of splitting on each observed value in the in the predictor space *regardless of the time at which is was observed*. Therefore, covariate values in $\boldsymbol{\gamma} = [\boldsymbol{\gamma}(t_1), \ldots, \boldsymbol{\gamma}(t_J)]'$ as well as any values of functionals $\mathbf{Z} = [\mathbf{Z}(t_1), \ldots, \mathbf{Z}(t_J)]'$ are all included in the predictor space and assessed as possible splitting values. Covariate values in $\boldsymbol{\gamma} = [\boldsymbol{\gamma}(t_1), \ldots, \boldsymbol{\gamma}(t_J)]'$ as well as any values of functionals $\mathbf{Z} = [\mathbf{Z}(t_1), \ldots, \mathbf{Z}(t_J)]'$ are all included in the predictor space and assessed as possible splitting values.

For a potential split, s_h, at a node, h, Bacchetti and Segal [9] propose to use a two sample rank statistic from the Tarone-Ware or Harrington-Fleming class as the splitting criterion

$$G(s_h) = \frac{\sum_{j=1}^{J} w_j[x_j - E_0(X_j)]}{\sqrt{[\sum_{j=1}^{J} w_j^2 Var_0(X_j)]}}, \tag{7.1}$$

where w_j is the weight for time t_j and

$$x_j = \sum_{i \in \mathcal{R}_j} I(\delta_i(t_j) = 1 \wedge z_i(t_j) \leq s_h), \tag{7.2}$$

which is interpreted as the number of individuals in the risk set at time t_j with an event at time t_j, as well as a repeatedly measured covariate value associated with time t_j that less than or equal to the potential splitting value s_h. Typical values for w_j are 1, n_j, or $\sqrt{n_j}$. The expected value and variances under the null hypothesis of no difference in the survival distributions between the two nodes created by s_h at time t_j,are defined as

$$E_0(X_j) = \frac{m_{j1} n_{j1}}{n_j} \tag{7.3}$$

and

$$Var_0(X_j) = \frac{m_{j1} n_{j1} (n_j - m_{j1})(n_j - n_{j1})}{(n_j - 1) n_j^2}, \tag{7.4}$$

where $m_{j1} = \sum_{i \in \mathcal{R}_j} I(z_i(t_j) \leq s_h)$, $n_{j1} = \sum_{i \in \mathcal{R}_j} \delta_i(t_j)$, and $n_j = R_j$. The split selected to divide the data, s_h^*, is the split that results in the largest between-node separation, as defined by equation (2.13).

An additional consideration in selecting the split is to determine whether it will result in minimum number of subjects in each resulting node. This restriction helps to ensure better tree stability, because when there are too few individuals in a node, patient risk will not reliably estimated. When repeatedly measured covariates are considered, we need to assess the number of subjects and pseudo-subjects that would result from a particular split and only consider those splits that would allow the minimum number in each node. These pseudo-subjects are created in the manner discussed by Bacchetti and Segal in subsection 2.4.6.2.

7.2 SIMULATION

7.2.1 Generating Covariate Data

We assess the accuracy of Bacchetti and Segal's time-dependent TSSA methodology when covariates are non-monotonically increasing with time. We present five different scenarios, referred to as models I-V. For each model, we simulate a baseline measurement and additional follow-up measurements at a set of discrete time points t_j, $j = 1, \ldots, J$. At each these discrete time points, t_j, the repeatedly measured covariate, $\mathbf{W}(t_j)$, is observed. For each individual, i, the event, T_i, or censoring, C_i, is also observed at one of these time points, such that the observed outcome is denoted by T_i^*.

We use a random effects model to generate the repeatedly measured covariate, V_3. The random effects for the i^{th} individual are denoted $\mathbf{b}_i = [b_{i0}, b_{i1}]'$, with $\mathbf{b}_i \sim N_2(\mathbf{0}, \mathbf{G})$. The between-subject variance matrix, $\mathbf{G}$, consists of the variance associated with the intercept, σ_1, the variance associated with time, σ_2, and the intercept-time covariance, σ_{12}.

We generate the repeatedly measured covariate, V_3, for the i^{th} individual as $V_{i3} \sim N_J(\mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\mathbf{b}_i, \boldsymbol{\Gamma}_i)$. The fixed and random design matrices for the intercept and time,

Table 7.1: Simulated covariates

Covariate	Distribution
Model Covariates	$V_1 \sim N(\mu_{v_1}, \sigma_{v_1})$
	$V_2 \sim Bin(p)$
	$V_3 \sim N_J(\mathbf{X}\boldsymbol{\beta} + \mathbf{Zb}, \boldsymbol{\Gamma})$
Noise Covariate	$V_4 \sim N(\mu_{v_4}, \sigma_{v_4})$
Event	$T \sim Exp(\lambda_h)$
Censoring	$C \sim U(S)$
Observed Outcome	$T^* = min(T, C)$

$\mathbf{X}_i$ and $\mathbf{Z}_i$, are both $J \times 2$ matrices consisting of a column of ones for the intercept, and a column of values $1, 2, \ldots, J$ for the observation times. Furthermore, $\boldsymbol{\Gamma}_i = \mathbf{Z}_i\mathbf{G}\mathbf{Z}_i' + \sigma_e\mathbf{I}_J$ is the variance-covariance matrix for individual i, and $\boldsymbol{\beta} = [\beta_0, \beta_1]'$ are the fixed effects parameters.

Specific parameter values utilized for each model are shown in Table 7.2; plots of the repeatedly measured covariate values over time are shown in Figure 7.1. Distributions of additional baseline covariates, which were generated in the same manner for each model, are shown in Table 7.1.

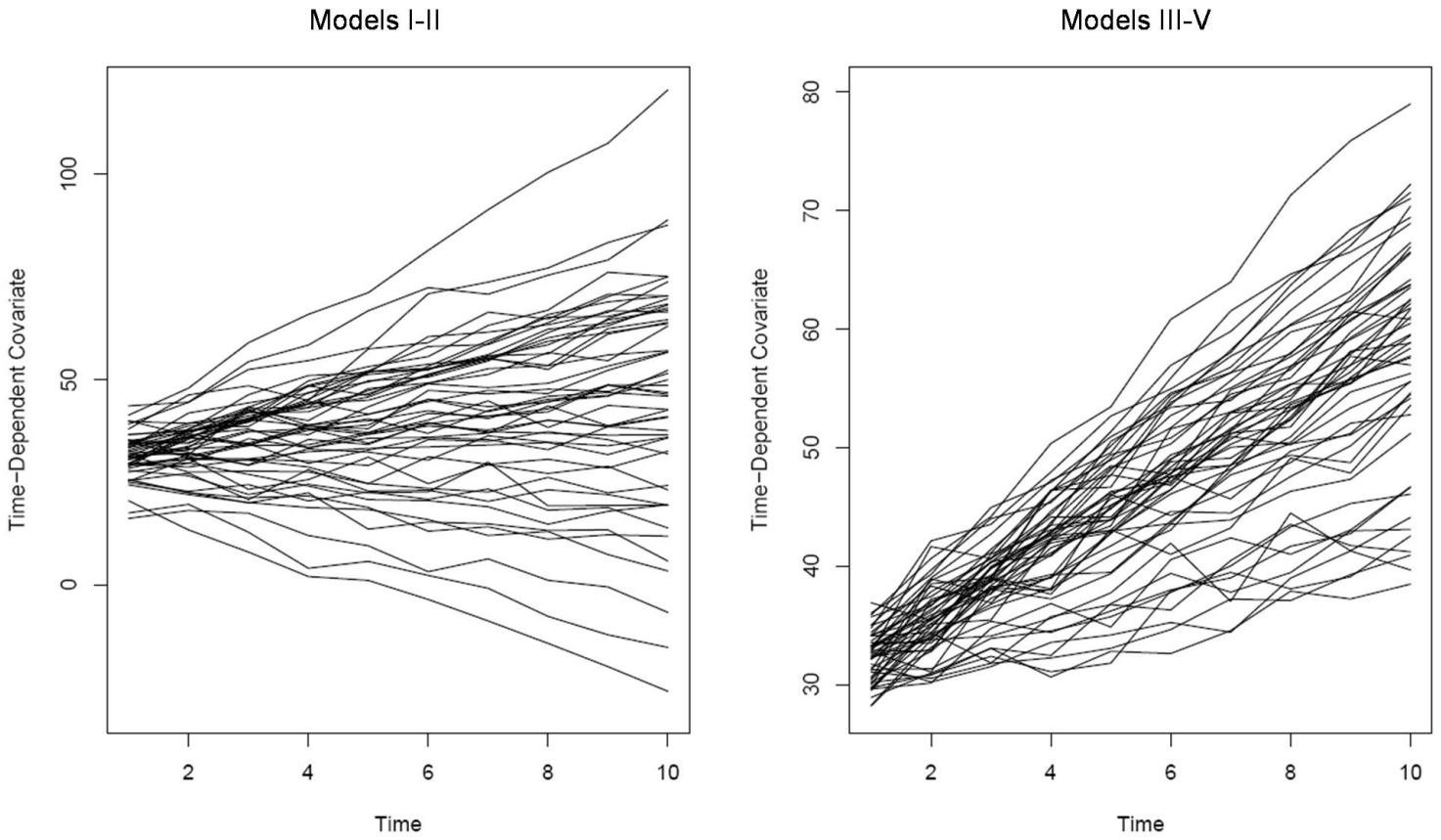

Figure 7.1: Time-Dependent Covariate V_3

Table 7.2: Parameter values

	Model				
Parameter	I	II	III	IV	V
V_3					
$\beta = \{\beta_0, \beta_1\}$	$\{30, 1.5\}$	$\{30, 1.5\}$	$\{30, 3\}$	$\{30, 3\}$	$\{30, 3\}$
$\gamma_G = \{\sigma_1, \sigma_2, \sigma_{12}\}$	$\{3, 3, 2\}$	$\{3, 3, 2\}$	$\{.5, .5, .5\}$	$\{.5, .5, .5\}$	$\{.5, .5, .5\}$
σ_e	5	5	2	2	2
J	10	10	10	6	6
V_1, V_2, V_4					
$\{\mu_{v_1}, \sigma_{v_1}\}$	$\{2, 4\}$	$\{2, 4\}$	$\{2, 4\}$	$\{2, 4\}$	$\{2, 4\}$
p	.5	.5	.5	.5	.5
$\{\mu_{v_4}, \sigma_{v_4}\}$	$\{4, .5\}$	$\{4, .5\}$	$\{4, .5\}$	$\{4,.5\}$	$\{4,.5\}$
Splits	(If "yes", send case to left node)				
h_1	is $V_3 < 30$?	is $V_3 \leq 40$?	is $V_3 \leq 40$?	is $V_3 \leq 40$?	is $V_3 \leq 35$?
h_2	is $V_2 = 0$?	is $V_2 = 0$?	is $V_2 = 0$?	is $V_2 = 0$?	is $V_2 = 0$?
h_3	is $V_1 \leq 10$?	is $V_1 \leq 10$?	is $V_1 \leq 10$?	is $V_1 \leq 10$?	is $V_3 \leq 45$?
Outcome					
$\lambda_h = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$	$\{.2, .6, 1, 3\}$	$\{.2, .6, 1, 3\}$	$\{.2, .6, 1, 3\}$	$\{.2, .6, 1, 3\}$	$\{.2, .6, 1, 3\}$
$S(CensoringTime)$	10	10	10	6	6

7.2.2 Generating Outcome Data

The basic structure for determining the distribution of the time-to-event outcome is shown in Figure 7.2, with specific cutoff values for each model displayed in Table 7.2. The first split is on the repeatedly measured covariate, such that individuals with higher values of covariate V_3 are at higher risk for the event. However, each individual's covariate V_3 changes over time, and consequently an individual's risk may also change over time.

To simulate these outcome data, we create pseudo-subjects for each individual based on whether the repeatedly measured covariate is above or below the splitting at each time point t_j. We employ both pseudo-subject creation methods proposed by Bacchetti and Segal [9]. For method one, a new pseudo-subject is created each time that the i^{th} subject's repeatedly measured covariate crosses the splitting value. Therefore, if the repeatedly measured covariate crosses the splitting value at $K-1$ distinct time points, they are divided into K pseudo-subjects, i_k, $k = 1, \ldots, K$. For method two, we only allow two pseudo-subjects. Thus, pseudo-subject one is assigned to all time points when the time-dependent covariate is less than or equal to the splitting value, and pseudo-subject two is assigned to all time points when the time-dependent covariate is above the splitting value. We denote the interval of time for which each pseudo-subject is followed by τ_{i_k}, such that $\sum_k^K \tau_{i_k} = t_J$, where t_J is the total follow-up time for each individual i.

We classify each pseudo-subject, i_k, into a terminal node, h, based on their corresponding repeatedly measured and baseline covariate observations. Because the pseudo-subject was created based on the value of the repeatedly measured covariate, the pseudo-subject i_k will be classified into only one node terminal node, h, throughout its follow-up time τ_{i_k}. Each pseudo-subject, i_k, is assigned hazard rate, λ_h, based on the terminal node into which they were classified. The event time for each pseudo-subject, i_k, is then generated as $T_{i_k} \sim \exp(\lambda_h)$. One additional unit of time is added to each event time, T_{i_k}, to ensure that no pseudo-subjects had an event until after the first observation. If $T_{i_k^*} \leq \tau_{i_k^*}$ for a particular pseudo-subject i_k^*, the failure for individual i occurs at time $\sum_{k=1}^k \tau_{i_k^*}$ and all subsequent pseudo-subjects are unobserved. However, if $T_{i_k^*} > \tau_{i_k^*}$, pseudo-subject i_k^* is censored and the observation time for pseudo-subject i_{k^*+1} begins. If an individual, i, reaches time t_J and

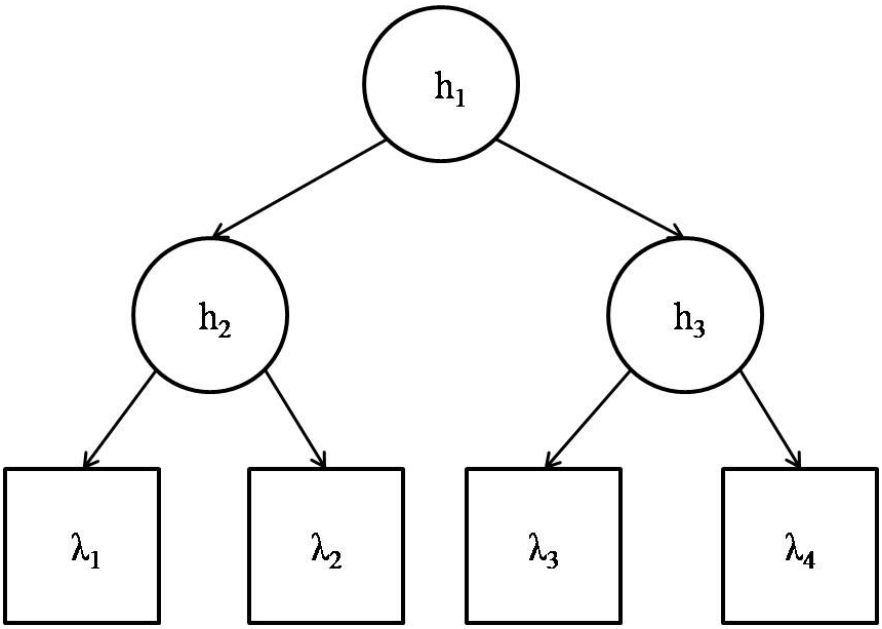

Figure 7.2: Tree structure

none of its pseudo-subjects i_k, $k = 1, \ldots, K$, have had an event, the final pseudo-subject i_K is censored at time t_J, and thus the individual i is also censored at time t_J.

7.2.3 Computing Methods

We wrote code in *R* [29] to generate all simulation data (see Appendix A.2). We also wrote our own programs in R to grow and prune Bacchetti and Segal's [9] time-dependent tree-structured survival models with the two-sample log-rank statistic. To grow the trees, we utilized a minimum node size of 20 subjects (or pseudo-subjects) and required at least 60 subjects (or pseudo-subjects) in a node to consider splitting. We utilized Bacchetti and Segal's first pseudo-subject method, which only allows for a maximum of two unique pseudo-subjects for each individual. In models I, III, and IV we also utilized LeBlanc and Crowley's [26] split-complexity pruning algorithm based on an independent test sample of $N = 200$ and a complexity parameter of $\alpha = 3$ to select a subtree.

7.2.4 Model Criteria

For all models, we grew the trees to the largest possible size, and then assessed their accuracy using three accuracy criteria:

(1) The percentage of models that split on the correct covariate at each node shown in Figure 7.2.

(2) The mean bias and empirical standard deviation (ESD) of the splitting statistic at nodes h_1 and h_3.

(3) The percentage of models that selected the correct covariates at all four nodes.

For criteria (1), the percentage correct was conditional on the model reaching the designated node of interest through the correct covariates, and denominator values were adjusted accordingly. For criterion (2), the mean and ESD were taken only from those models that correctly reached the node of interest and then correctly selected the covariate to split that node.

For the pruned trees based on models three and four, we also summarize the quartiles, mean, and standard deviation of the final number of terminal nodes. The true number of terminal nodes in each model was four.

7.2.5 Model Results

Model I was generated so that the data so that the first split was on the repeatedly measured covariate at a value of $V_3 \leq 30$, as shown in Figure 7.2. This value was observed for many individuals at baseline as well as during follow-up. At the first split, all three methods performed very well. However, each method decreased in accuracy further down the tree. The baseline model produced the least biased splitting values when the splitting covariate was correctly selected. With respect to overall tree accuracy, using the baseline covariate, instead of the repeatedly measured covariate, produced a slightly more accurate model.

For model II, we kept all parameters the same as in model I, except for increasing the true cutoff value of the repeatedly measured covariate from 30 to 40. The majority of individuals had observed V_3 values less than 40 at baseline. Thus, the baseline model

Table 7.3: Accuracy results for models I-V

		Node h_1		Node h_2	Node h_3		All Nodes
Model	Method	% Correct	Bias(ESD)	% Correct	% Correct	Bias(ESD)	% Correct
	PS-1	100	2.36(1.32)	64.8	82.6	-.33(.97)	53.5
I	PS-2	100	2.26(1.32)	64.6	82.5	-.33(.98)	53.2
	Baseline	99.8	1.39(1.14)	68.6	83.7	-.19(1.12)	57.0
	PS-1	70.6	4.90(1.45)	99.43	40.23	-.43(1.15)	28.1
II	PS-2	70.6	4.90(1.45)	99.43	40.93	-.55(1.50)	28.6
	Baseline	4.9	-2.99(3.56)	87.76	18.37	-.82(1.26)	.6
	PS-1	94.9	3.85(.78)	91.25	77.45	-.45(.97)	66.7
III	PS-2	94.9	3.85(.78)	91.25	77.34	-.45(.97)	66.6
	Baseline	9.1	-8.05(1.60)	46.15	49.45	.62(2.18)	1.2
	PS-1	84.8	4.09(.81)	81.13	81.01	-.37(.98)	55.8
IV	PS-2	84.8	4.09(.81)	81.13	81.11	-.37(.98)	55.9
	Baseline	6.9	-6.95(1.80)	68.12	27.54	-.67(1.57)	1.2
	PS-1	98.6	4.51(3.34)	25.15	78.40	.13(2.31)	20.9
V	PS-2	98.6	4.51(3.34)	25.15	78.40	.13(2.31)	20.9
	Baseline	98.4	.07(.76)	34.25	15.55	8.98(.50)	6.2

Table 7.4: Number of terminal nodes in pruned tree

Model	Method	Mean(ESD)	25^{th} Percentile	50^{th} Percentile	75^{th} Percentile
I	PS-1	11.94(4.04)	9	12	15
	PS-2	12.12(4.26)	8	12	15
	Baseline	7.30(2.04)	6	7.50	9
III	PS-1	12.21(6.71)	7	11	17
	PS-2	12.25(6.80)	7	11	17
	Baseline	5.39(2.57)	3	5	7.25
IV	PS-1	12.10(6.40)	7	10.5	16
	PS-2	12.13(6.44)	7	11	16.25
	Baseline	5.67(2.64)	3	5	8

was only able to correctly select covariate V_3 as the first split in 4.9% of the simulated datasets. For the repeatedly measured methods, the splitting values for this model were generally the most biased. Overall, none of the methods performed particularly well. The two pseudo-subject methodologies only provided completely accurate trees for about 28% of the simulated datasets, and the baseline method only provided completely accurate trees for .6% of the datasets.

Due to the lower percentages of accuracy in model II, for model III we kept the larger cutoff value of $V_3 \leq 40$, but decreased both the within-subject and between subject errors, as well as increased the random effect slope parameter (see Figure 7.1. The baseline models still performed very poorly, as they were not able to select the first split correctly due to the fact that so few individuals had an observed value of $V_3 > 40$ at baseline. However, the decrease in variance parameters resulted in the most accurate time-dependent tree models overall, selecting the entirely correct tree for approximately 66% of the simulated datasets. For this model, we also assessed the ability of the pruning methodology to determine the

correct number of terminal nodes. Out of 500 simulations, the time-dependent covariate methods both had a median of 11 terminal nodes, which is almost double the true size of the tree. The baseline-only method had a median of 5 terminal nodes, which is only one terminal node larger than the number of terminal nodes in the true tree model.

In model IV, we kept the same parameter values as in model III but only utilized $J = 6$ follow-up measurements instead of $J = 10$ measurements. The result was a less accurate model, overall, than in model III. However, the results for the time-dependent covariate models were still better than in models I and II. We also assessed the pruning methodology with this model. The results were very similar to the pruning results from model III. The first pseudo-subject method, which only allows for a total of two-pseudo-subjects to be created, had a median of 10.5 terminal nodes. The second pseudo-subject method had a median of 11 terminal nodes. The trees using only baseline covariates had a median of five terminal nodes.

In model V, we utilized parameter values from model IV, but split on the repeatedly measured covariate two times: first on a value of $V_3 \leq 35$, and then on a value of $V_3 \leq 45$. This resulted in the least accurate model of the five, with the repeatedly measured methods only selecting the a completely accurate tree for 21% of the simulated datasets. However, because the initial split value of 35 was observed by some individuals at baseline, this model performed better than in model IV, dramatically increasing its accuracy at the first split. Of course, it was still not able to detect the split of $V_3 \leq 45$ for the third split because this value was not observed at baseline.

In summary, when the cutoff value for the time-dependent covariate was at a level observed at baseline, the models utilizing only the baseline covariates were more accurate. However, when the time-dependent covariate cutoff value was at a level not typically observed at baseline, the models utilizing the time-dependent covariates were much more accurate. Overall, smaller variance and covariance parameters and larger numbers of follow-up time points resulted in higher accuracy for the time-dependent models. Surprisingly, the two pseudo-subject methods performed very similarly, only differing by a few tenths of a percentage point, if at all.

Table 7.5: Models

Model #	Anxiety Functional(s) Included in the Model	Description
1	$Z(t_j) = \gamma(t_1)$	Baseline anxiety
2	$Z(t_j) = \gamma(t_j)$	time-dependent anxiety
3	$Z_1(t_j) = \frac{\gamma(t_j)-\gamma(t_{j-1})}{t_j-t_{j-1}}$	Weekly Rate of Change of Anxiety
	$Z_2(t_j) = \gamma(t_1)$	Baseline Anxiety
4	$Z(t_j) = \gamma(t_j) - \gamma(t_1)$	Anxiety Change from Baseline
5	$Z_1(t_j) = \gamma(t_j) - \gamma(t_1)$	Anxiety Change from Baseline
	$Z_2(t_j) = \gamma(t_1)$	Baseline Anxiety

7.3 MTLD-I APPLICATION

In the acute phase of MTLD-I, individuals had their anxiety levels observed at week one, as well as each follow-up week. Using these data, our goal was to create prognostic models based on clinical characteristics, including temporal features of anxiety. To create our models, we utilized Bacchetti and Segal's time-dependent TSSA methodology, requiring at least 20 subjects (or pseudo-subjects) in each terminal node and a complexity parameter of $\alpha = 2$. To prune the models, we utilized LeBlanc and Crowley's [26] proposed bootstrapping method, described in Section 2.4.4.2.

In total, we generated five different models, each utilizing baseline covariates shown in Table 4.1 as well as functionals of time-dependent anxiety. The functionals utilized in each model are shown in Table 7.5.

7.3.1 Tree-Structured Survival Models

Model 1 The model created with week one anxiety is shown in Figure 7.3. The first variable to split the data is baseline anxiety. For those individuals with higher baseline anxiety, the next split is on baseline depression. The Kaplan Meier plots of each terminal node in the

model show that individuals with higher depression and higher anxiety do not respond to treatment as quickly as other individuals. *Model 2* The model created with time-dependent anxiety covariate is shown in Figure 7.4. As in the baseline model, the first split is on anxiety $\leq .417$. However, for those individuals with higher anxiety, the next split is on anxiety $\leq .583$. For individuals with highest anxiety, self esteem is the next split. The Kaplan Meier plot in Figure 7.5 shows that individuals with high ($> .583$) anxiety and low (≤ 5.5) self-esteem have the worst prognosis. However, individuals with high anxiety and high self-esteem have a better prognosis than individuals with medium ($> .416$ but $\leq .538$) anxiety. *Model 3* The model including the anxiety rate of change functional as well as baseline anxiety was the same as the model utilizing only baseline anxiety. These results are shown in Figure 7.3. *Model 4* The model including the change from baseline functional is shown in Figure 7.6. The resulting tree model shows that the individuals with baseline HDRS> 20.5 and an increase in anxiety of more than 1.65 from baseline is associated with the higher risk of not responding to treatment. However, if an individual's baseline anxiety increases more than 1.65 but their baseline depression score was less than 20.5, they have a much better prognosis. *Model 5* The model including both the change from baseline functional and also baseline anxiety is shown in Figure 7.7. This model identifies a group of individuals who are highly at risk for not responding to treatment for anxiety. This group is identified by individuals who have low self esteem (ISEL-SE ≤ 5.5), baseline anxiety above .917, and an increase in anxiety of at least .67 from baseline. Individuals with anxiety increase no more than .67 from baseline (or who decrease in anxiety) have a much better prognosis.

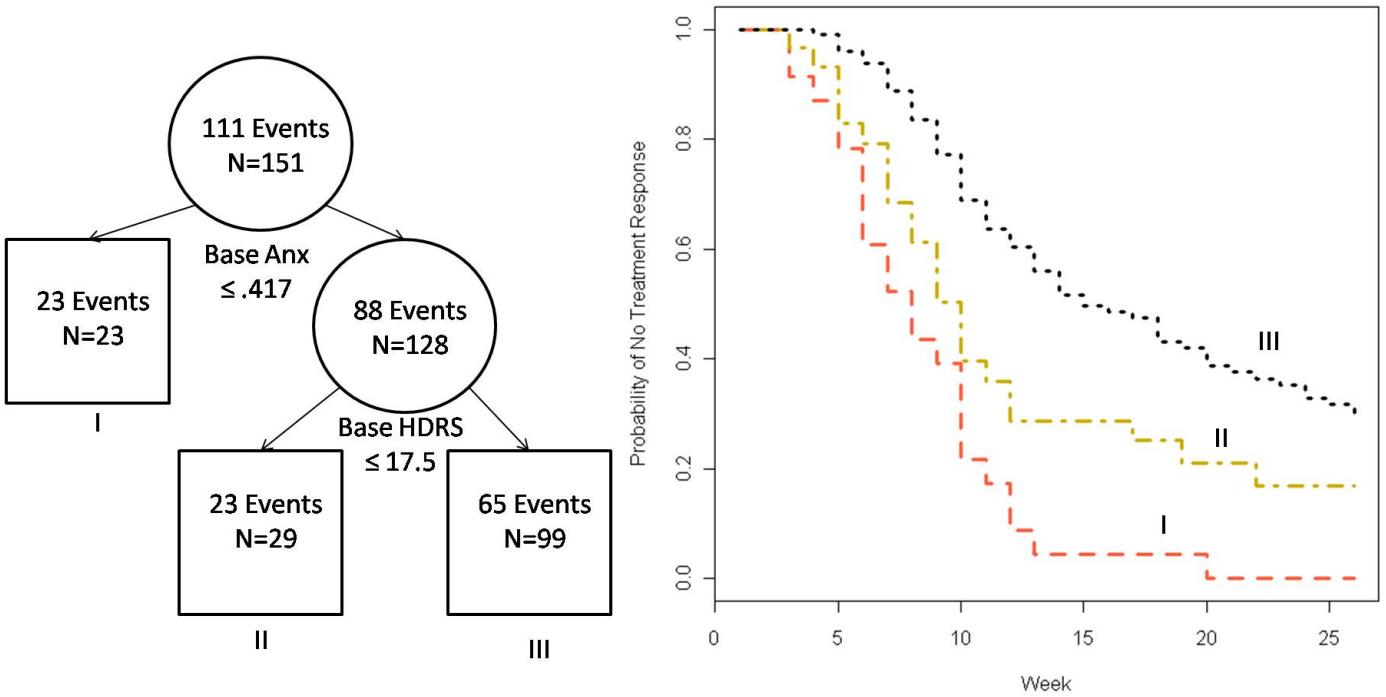

Figure 7.3: Tree model using only baseline anxiety

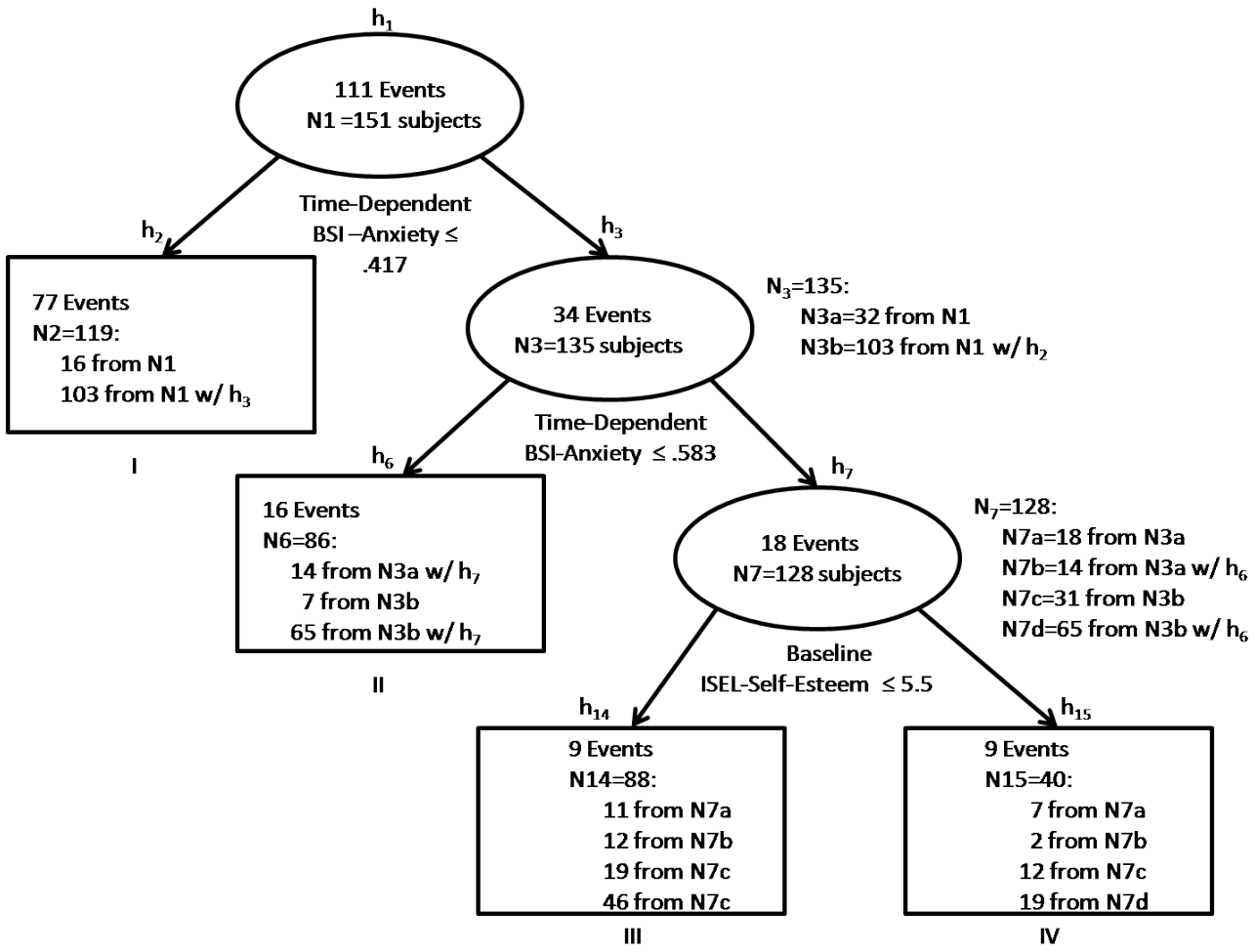

Figure 7.4: Detail of pseudo-subjects for time-dependent anxiety model

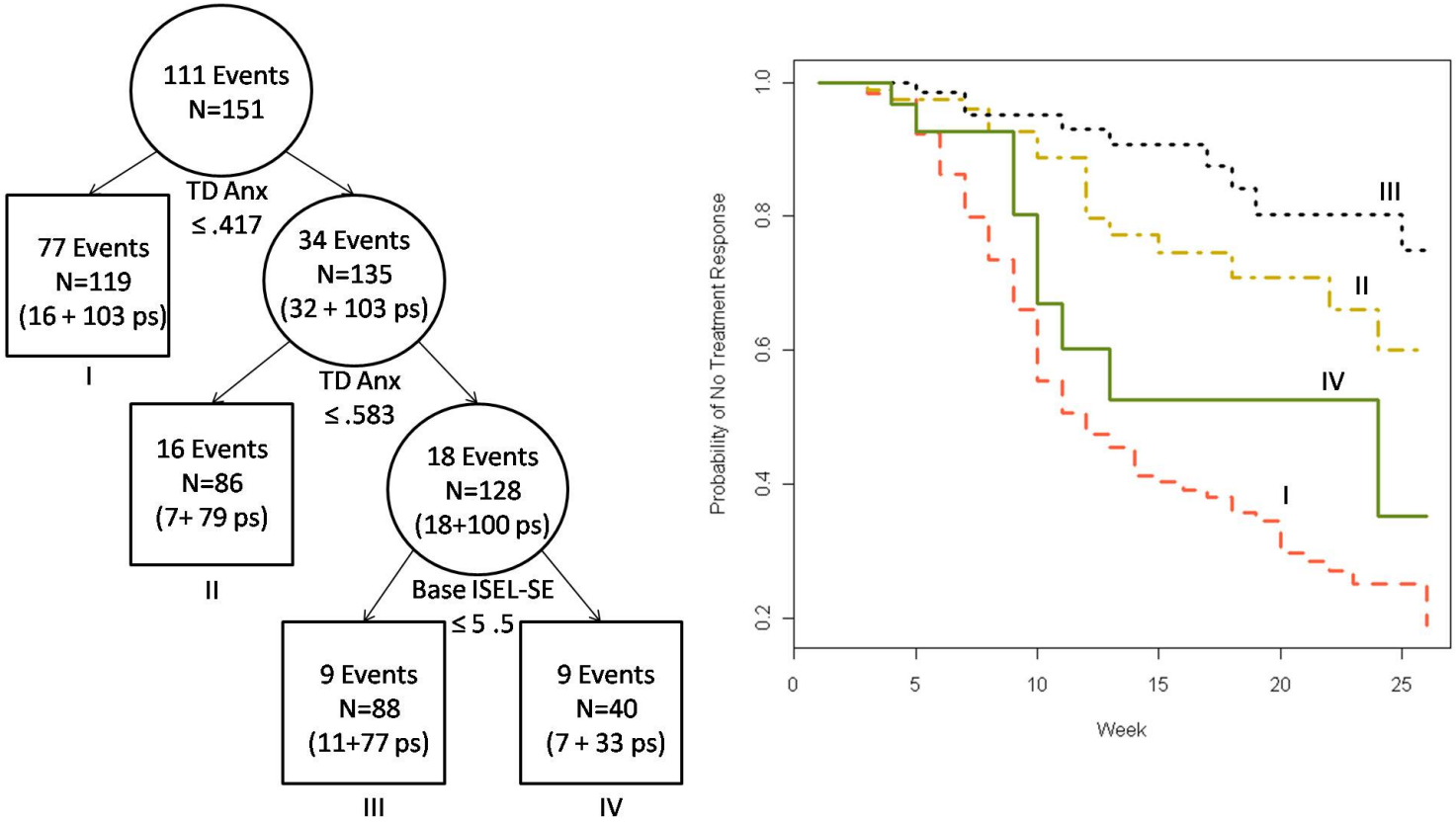

Figure 7.5: Time-dependent anxiety model

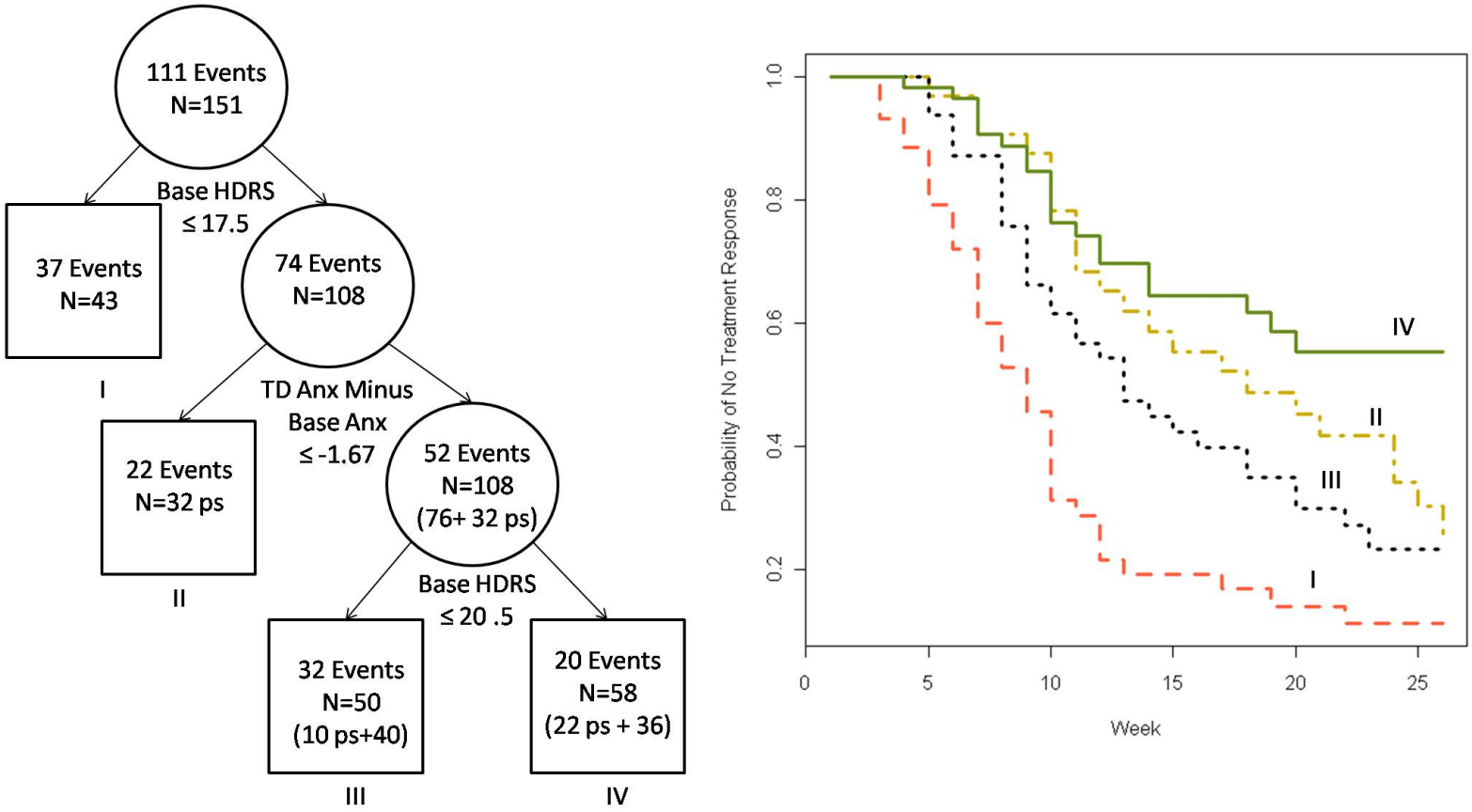

Figure 7.6: Time-dependent anxiety change from baseline model

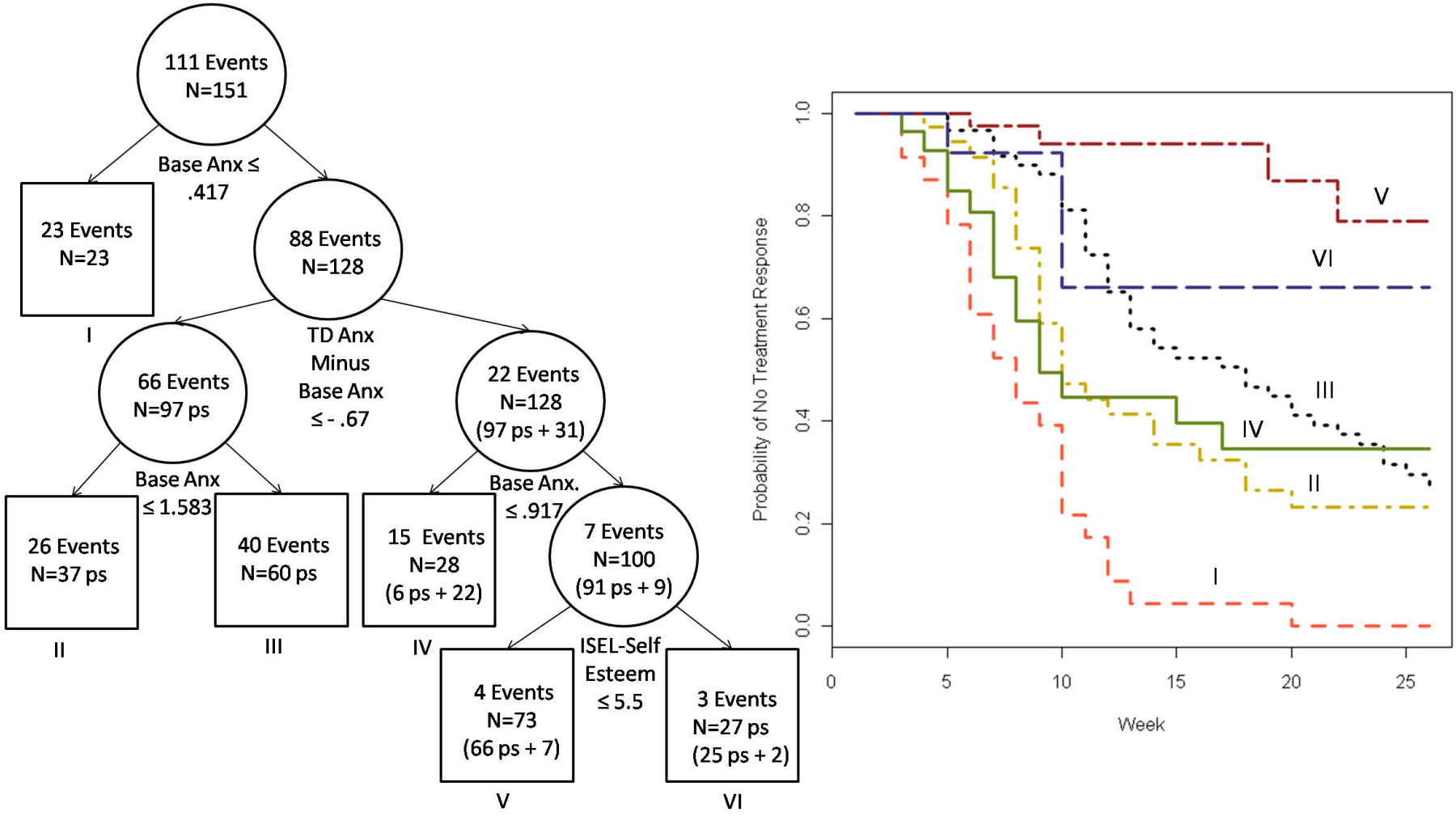

Figure 7.7: Time-dependent anxiety change from baseline model (including baseline anxiety)

7.3.2 Model Assessment

We used to Empirical Brier Score to assess the predictive abilities of the four unique models created with the MTLD-I data. As in Chapter 6, we used a jackknife approach to assess each model's predictive ability. Thus, for each model, we created subsets L^{-i}, $i = 1, \ldots, N$, by iteratively removing the i^{th} individual from the set including entire sample, $\mathcal{L}$. Using individuals in a subset $\mathcal{L}^{-i}$, we first grew a survival tree based on Bacchetti and Segal's time-dependent TSSA method [9], and then pruned the survival tree based on LeBlanc and Crowley's bootstrapping method [26]. The resulting tree is denoted by H^{-i}.

To estimate the Brier score contribution for the excluded individual i at a time t_j, we classified the individual into a terminal node using their covariate values assigned to time t_j, and estimated the Brier score based on the survival and censoring distributions of their assigned terminal node [38]. For baseline covariates, we associated the baseline value with each failure point t_j, $j = 1, \ldots, J$. This process was repeated for each subset $\mathcal{L}^{-i}$, $i = 1, \ldots, N$, and the resulting contributions were averaged together at each time point. We also averaged all time points together to obtain the integrated Brier score.

Figure 7.8 displays the relative predictive accuracies of our models when used repeatedly throughout the course of the 26 week treatment. Interestingly, this figure shows that the model using baseline anxiety was most accurate in predicting survival over the course of treatment. The two models utilizing change from baseline are relatively accurate during the first ten weeks of treatment, however, after this time period they hover at approximately .25, which corresponds to the value obtained by chance alone. We also note the sudden increase and subsequent decrease in the Brier score for the baseline plus change-from-baseline model. The model that performed worst was the time-dependent anxiety model. After about week 10 the time-dependent anxiety model does very poorly, almost inversely predicting the survival outcome. The integrated Brier scores for both the entire course of treatment and well as only the two first months in which individuals can respond to treatment are shown in Table 7.6. Models incorporating baseline anxiety only have the lowest integrated Brier scores, implying that they are most accurate when utilized for prediction throughout the course of treatment.

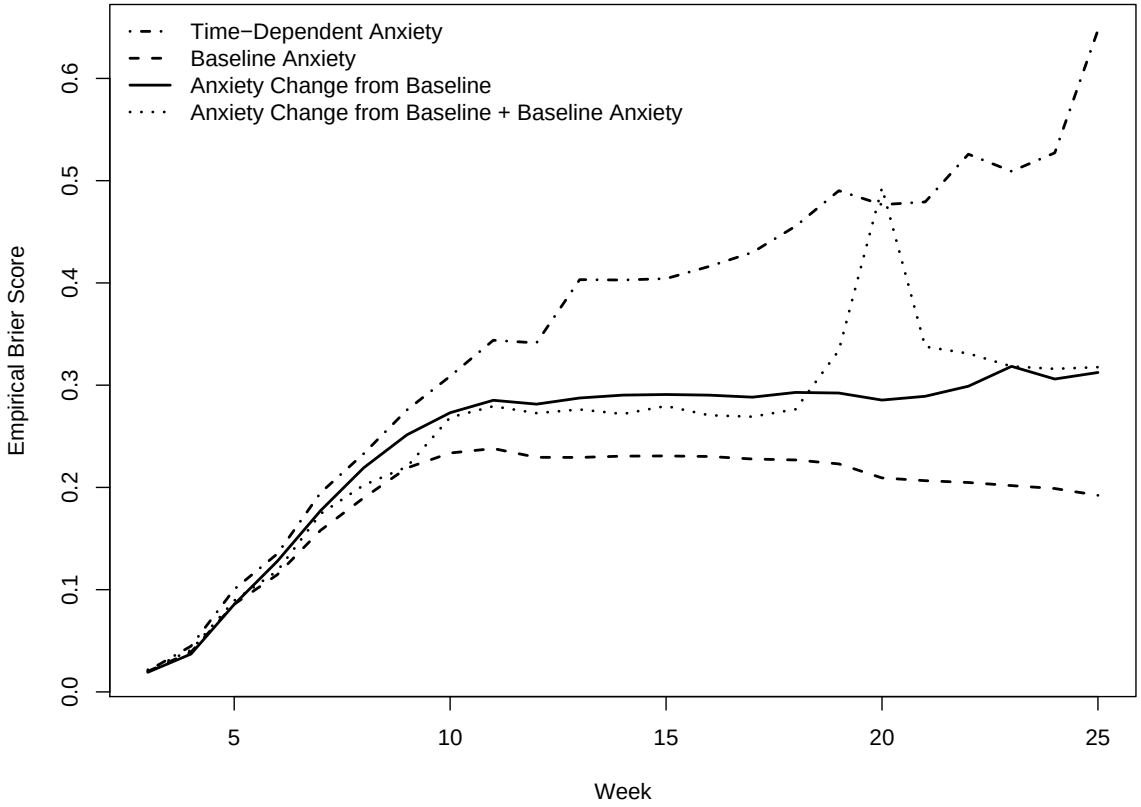

Figure 7.8: Empirical Brier scores for weeks 3 through 26 of acute treatment

Table 7.6: Integrated Brier scores

Form(s) of Anxiety in Tree Model	Weeks 3-10	Weeks 3-26
Baseline	.133	.189
Time-Dependent	.164	.355
Change from Baseline	.145	.243
Change from Baseline, Baseline	.142	.251

7.4 DISCUSSION

Our simulation study results show that time-dependent TSSA can be valuable in situations where the "true" cutoff value may not be observed until later in follow-up. However, there are still a number of issues that need to be addressed before time-dependent TSSA can be used confidently. One of the biggest issues is the resulting tree size. Even though pruning may help to keep the tree size in check when non-monotonically increasing or decreasing repeatedly measured covariates are used, our simulation studies show that both pseudo-subject methods result in models much larger than if only baseline covariates are used.

One possible solution, when assessing the size of a node to determine whether it should continue to be slit upon, is to weight each subject (or pseudo-subject) based on the proportion of their follow-up that they represent. For example, a pseudo-subject followed for only two out of their total ten observed follow-up time points would add 2/10 of a unit to the node size. An individual not broken into pseudo-subjects, however, would add one whole unit to the node size. This method of counting the number of individuals in each node would help to keep the overall number of nodes created by the survival tree at a more manageable level and also result in less time spent growing the tree.

Figure 7.8 provides interesting results regarding the accuracy of the tree-structured models when used throughout treatment to predict outcome. Particularly surprising was the poor predictive ability of the repeatedly measured anxiety model. However, after reflecting upon these results further, they begin to make more sense. Consider an individual who is relatively

difficult to treat, and therefore does not even begin to improve until late in the treatment process (time t_j). When this individual finally begins to improve, their anxiety scores begin to decrease. However, when these updated anxiety measurements are used at time t_j to classify the individual into the TSSA model to estimate its predictive value, the individual will be classified into a group associated with lower risk due to their lower anxiety scores. As a result, the individual will be incorrectly predicted to have an event prior to time t_j. However, in reality, the individual's lower anxiety scores at time t_j do not imply shorter survival time altogether, but rather shorter *residual* survival time after t_j. Therefore, if the current repeatedly measured Brier score statistic were altered to predict residual survival, as opposed to overall survival, we might see very different and more meaningful results.

8.0 CONCLUSIONS

The methodology in this dissertation is a significant contribution to the field of tree-structurd survival analysis. We first presented a stochastic multiple imputation method method for accommodating missing covariate data in tree-structured survival analysis. Our missing data methodology is particularly useful for tree-structured analysis and an improvement over current methods because it allows the user to relax assumptions regarding linear covariate relationships while still maintaining flexibility, simplicity and accuracy at a reasonable level. Through simulation studies, our methodology was shown to be as effective or more effective than other current methods that have been proposed. We also proposed two methods for incorporating temporal features of repeatedly measured covariates into tree-structured survival analysis. Through the Brier score, we were able to show that utilizing information from repeteadly measured covariates can greatly improve the ability of the model to accurately estimate risk.

For a public health application, we used the MTLD data [1] to develop multiple algorithms that could be used both at baseline and throughout treatment to classify individuals into risk groups. In these models, we consistently found that the covariate splitting value most strongly related to the time-to-event outcome was baseline BSI anxiety $\leq .416$. However, after this similarity, the applications of our proposed methods resulted in a number of different tree models that varied in size as well as covariate and splitting value selection. One cause of this variability was the different underlying assumptions that were utilized in each of the models. Specifically, we did not use the same splitting statistic, pruning algorithm, or parameter values for each tree model, which made it more challenging to directly compare the predictive abilities of each of the models and subsequently select one final model.

In the future, it would be useful to be able to identify one single model that could be

reliably used in a clinical setting. To do this, first it would be necessary fit a series of models based on the same set of assumptions; specifically, using the same splitting statistic and parameter values. Next, we would need to assess each of these models based on the same criterion. Throughout this dissertation, we used the Brier Score to assess the predictive abilities of many of our models. However, as discussed previously in Section 7.4, this may not be the most ideal measure with respect to assessing the time-dependent TSSA models due to the fact that it utilizes overall survival instead of residual survival. In order to select an overall "best" tree, we would need to develop a statistic that will assess the predictive abilities of each model when utilized throughout the course of treatment. In addition, a cross validation study utilizing on a much larger and independent dataset would be extremely helpful in the final selection of one model that could be used in clinical practice.

APPENDIX

R CODE

A.1 MISSING COVARIATE DATA

```
library(survival)
library(rpart)

##CREATE MULTIPLE IMPUTATIONS##
impute.mi<-function(form,dat7,colm,B){
##form: formula for imputation (ex: "Ymis~X1+X2+X3")
##dat7:dataset
##colm: column with missing observations
##B: number of MIs

##set up complete data and and grow tree##
form2<-as.formula(form)
datai<-dat7
impvar<-dat7[,colm]
fit<-rpart(form2,datai[is.na(datai[,colm])==FALSE,],maxsurrogate=0)

##predict missing obs##
n<-datai[is.na(dat7[,colm])==TRUE,] ##n=missing data##
```

```
pred<-predict(fit,n) ##get predicted values##
pr<-as.vector(pred)
miss<-as.numeric(as.vector(names(pred)))  ##row names for the missing observations##

##get the standard deviation of the residuals in each terminal node##
tns<-unique(fit$where)  ##terminal node names##
sdf<-matrix(c(0),nrow=length(tns))
mnf<-matrix(c(0),nrow=length(tns))
sdall<-matrix(c(0),nrow=length(pr))
obs<-dat7[is.na(dat7[,colm])==FALSE,colm]  ##observed data--counterpart of "n"##
for(f in 1:length(tns)){
mnf[f]<-mean(fit$y[fit$where==tns[f]])
obs2<-obs[fit$where==tns[f]] ##observed data in terminal node "f"##
sdf[f]<-sd(obs2-mnf[f])  ##subtract mean from each observed data point in
    ##node "f" and get sd##
sdall[pr==mnf[f]]=sdf[f]  ##record sd the predicted values equal the mean of the node##
}

##impute the mean + the error into the obs with missing data##
impdata<-matrix(c(0),nrow=nrow(datai),ncol=B)
for(b in 1:B){
impdata[,b]=impvar
dev1<-rnorm(length(sdall),0,sdall)
impdata[is.na(impdata[,b])==TRUE,b]=pr+dev1
}
cns<-"i1"
if(B>1){
for(b in 2:B){
cns2<-paste("i",b,sep="")
cns<-c(cns,cns2)
```

```
}
}
colnames(impdata)<-cns
return(impdata)
}


###FIND BEST SPLIT BASED ON LIKELIHOOD RATIO STATITSIC###
lr.split<-function(data99,covs,tcol,dcol,minbucket, minsplit){
##data99: dataset
##covs: column placement of covariates to be tested
##tcol: column with the event times
##dcol: column with indicators for events (event=1, censor=0)
##minbucket: minimum number of obs in a terminal node
##minsplit: minimum number of obs in a node to allow it to split

if(nrow(data99)>minsplit){
splitvar=-99
splitval=-99
Nleft=-99
Nright=-99
stat2=-99
for(k in covs){
datasort<-data.frame(data99[order(data99[,k]),])
split<-matrix(c(0),nrow=(nrow(data99)))
N<-nrow(data99)
for(p in (minbucket):(N-minbucket-1)){
split[datasort[,k]<datasort[p,k] | datasort[,k]==datasort[p,k]]=1
split[datasort[,k]>datasort[p,k]]=0
datasort2<-cbind.data.frame(datasort,split)
```

```
if(sum(datasort2$split)!=0 & sum(datasort2$split)!=N){
fit<-survdiff(Surv(datasort2[,tcol],datasort2[,dcol])~datasort2$split)
lrstat<-fit$chisq
if(abs(lrstat)>stat2 & datasort2[p,k]!=datasort2[p+1,k]){
  splitval<-(datasort2[p,k]+datasort2[p+1,k])/2
splitvar<-k
stat2<-lrstat
Nleft<-sum(datasort2$split)
Nright<-N-sum(datasort2$split)
}
}
}
}
out<-c(splitvar,splitval,stat2,Nleft,Nright)
return(out)
}
if(nrow(data99)<minsplit)return("Not enough observations to split")
}


###FIND BEST SPLIT BASED ON FULL LIKELIHOOD DEVIANCE STATITSIC###
fld.split<-function(dataf,covs,tcol,dcol,minb,mins){
##dataf: dataset
##covs: column numbers of covariates to be tested in "dataf"
##tcol: column of "dataf" containing the event time
##dcol: column of "dataf" containing the event indicator (event=1, censor=0)
##minb: minimum terminal node size
##mins: minimum obs needed to split a node

cns<-colnames(dataf)
```

```
cns2<-cns[covs]
cns1<-cns2[1]
if(length(cns2)>1){
for(i in 2:length(cns2)){
usef<-paste(cns2[i],cns1,sep="+")
}
}
if(length(cns2)==1)(usef=cns1)
use2<-paste("Surv(dataf[,tcol],dataf[,dcol])",usef,sep="~")
use3<-as.formula(use2)
treeo<-rpart(use3,data=dataf,minbucket=minb,minsplit=mins,cp=-1)

varo2<-as.character(treeo$frame[1,1])
cdf<-colnames(dataf)
for(i in 1:ncol(dataf)){
if(varo2==cdf[i])colnum=i
}
devo<-treeo$splits[1,3]
splo<-treeo$splits[1,4]
varo<-colnum
out<-c(varo,splo,devo)
return(out)
##list(varo=varo,splo=splo,devo=devo)
}

##GROW TREE WITH LR or FLD SPLIT STATISTIC AND MULTIPLE IMPUTATIONS##
grow.mi<-function(data,impcol,outB,usecovs,tcol,dcol,B,minbucket,minsplit,splitstat){

##data: dataset
##timecol: column that holds the event times
```

```
##dcol: column that holds the event indicator (1=event)
##impcol: column containing the missing covariate
##outB: output from impute.MI function (B vectors of imputed values)
##usecovs: col vector of nonmissingcovs to be considered
##B: number of multiple imputations
##minbucket: the minimum size of a terminal node
##minsplit: the minimum number of obs in a node to allow it to split
##splitstat: equals "fld" if FLD stat. Equals "lr" for LR stat

i=1
j=2
l=1
nodes<-matrix(c(0),nrow=nrow(data),ncol=10)
splits<-matrix(c(0),nrow=nrow(data),ncol=10)
vars<-matrix(c(0),nrow=nrow(data),ncol=10)
stats<-matrix(c(0),nrow=nrow(data),ncol=10)

time<-data[,tcol]
d<-data[,dcol]
data2<-cbind.data.frame(data,time,d)

##find best root node split for all data for covariates in vector 'use'###
if(splitstat=="fld")tempout<-fld.split(data2,usecovs,tcol,dcol,minbucket,minsplit)
if(splitstat=="lr")tempout<-lr.split(data2,usecovs,tcol,dcol,minbucket,minsplit)
devo<-tempout[3]
splo<-tempout[2]
varo<-tempout[1]

##find best root node split for each imputed vector##
bigdat<-cbind.data.frame(outB,data2)
```

```
dev<-matrix(c(0),nrow=B)
spl<-matrix(c(0),nrow=B)
if(splitstat=="lr"){
for(b in 1:B){
tempout2<-lr.split(bigdat,b,tcol+B,dcol+B,minbucket,minsplit)
dev[b]<-tempout2[3]
spl[b]<-tempout2[2]
}
}
if(splitstat=="fld"){
for(b in 1:B){
tempout2<-fld.split(bigdat,b,tcol+B,dcol+B,minbucket,minsplit)
dev[b]<-tempout2[3]
spl[b]<-tempout2[2]
}
}

##compare imputed vector stats of best stat from observed data##
pick<-matrix(c(0),nrow=B)
pick[dev>devo]=1
sums<-sum(pick)
if(sums>=B/2)var1=1
if(sums<B/2)var1=varo
if(sums>=B/2)split1=median(spl)
if(sums<B/2)split1=splo
if(sums>=B/2)stat1=median(dev)
if(sums<B/2)stat1=devo

##if the first split is not on the imputed covariate, proceed as normal##
if(var1!=impcol){
```

```
nodes[data2[,var1]<=split1,l]=j
nodes[data2[,var1]>split1,l]=j+1
splits[,l]=split1
vars[,l]=var1
stats[,l]=stat1
}

##if the first split IS on the imputed covariate,##
##determine which obs go to which node##
if(var1==impcol){
left<-matrix(c(0),nrow=nrow(data2),ncol=B)
for(b in 1:B){
left[outB[,b]<=split1,b]=1
}
sums<-rowSums(left)
nodes[sums>=B/2,l]=j
nodes[sums<B/2,l]=j+1
splits[,l]=split1
vars[,l]=var1
stats[,l]=stat1
}

##find best splits for descendant nodes##
i=2
j=4
l=1
sum=1
cont=1
while(cont==1){
if(i>sum){
```

```
l=l+1
sum<-sum+2^(l-1)
}

if(nrow(data2[nodes[,(l-1)]==i,])>(minsplit)){
if(splitstat=="lr")tempout<-lr.split(data2[nodes[,(l-1)]==i,],
usecovs,tcol,dcol,minbucket,minsplit)
if(splitstat=="fld")tempout<-fld.split(data2[nodes[,(l-1)]==i,],
usecovs,tcol,dcol,minbucket,minsplit)
devo<-tempout[3]
splo<-tempout[2]
varo<-tempout[1]

bigdat<-cbind.data.frame(outB[nodes[,(l-1)]==i,],data2[nodes[,(l-1)]==i,])
dev<-matrix(c(0),nrow=B)
spl<-matrix(c(0),nrow=B)
if(splitstat=="lr"){
for(b in 1:B){
tempout2<-lr.split(bigdat,b,tcol+B,dcol+B,minbucket,minsplit)
dev[b]<-tempout2[3]
spl[b]<-tempout2[2]
}
}
if(splitstat=="fld"){
for(b in 1:B){
tempout2<-fld.split(bigdat,b,tcol+B,dcol+B,minbucket,minsplit)
dev[b]<-tempout2[3]
spl[b]<-tempout2[2]
}
}
```

```
pick<-matrix(c(0),nrow=B)
pick[dev>devo]=1
sums<-sum(pick)
if(sums>=B/2)var1=1
if(sums<B/2)var1=varo
if(sums>=B/2)split1=median(spl)
if(sums<B/2)split1=splo
if(sums>=B/2)stat1=median(dev)
if(sums<B/2)stat1=devo

if(var1!=1){
nodes[data2[,var1]<=split1 & nodes[,(l-1)]==i,l]=j
nodes[data2[,var1]>split1 & nodes[,(l-1)]==i,l]=j+1
splits[nodes[,(l-1)]==i,l]=split1
vars[nodes[,(l-1)]==i,l]=var1
stats[nodes[,(l-1)]==i,l]=stat1
}
if(var1==1){
left<-matrix(c(0),nrow=nrow(data2),ncol=B)
for(b in 1:B){
left[outB[,b]<=split1 & nodes[,(l-1)]==i,b]=1
}
sums<-rowSums(left)
nodes[sums>=B/2 & nodes[,(l-1)]==i,l]=j
nodes[sums<B/2 & nodes[,(l-1)]==i,l]=j+1
splits[nodes[,(l-1)]==i,l]=split1
vars[nodes[,(l-1)]==i,l]=var1
stats[nodes[,(l-1)]==i,l]=stat1
}
```

```
}
i=i+1
j=j+2
kg<-table(nodes[,(l-1)])
kg2<-kg[rownames(kg)!="0"]
kg2[kg2>=(minsplit)]=9999
cont[sum(kg2)<9999]=0
}

last=0
cs<-colSums(nodes)
last<-length(cs[cs!=0])  ##number of layers##
use<-nodes[,1:(last)]
rownames(use)=seq(1:nrow(use))
use2<-unique(use)

splits2<-splits[,1:(last)]
splits3<-t(splits2[c(as.matrix(as.numeric(rownames(use2)))),])
vars2<-vars[,1:(last)]
vars3<-t(vars2[as.matrix(as.numeric(rownames(use2))),])
stats2<-stats[,1:(last)]
stats3<-t(stats2[as.matrix(as.numeric(rownames(use2))),])
tuse<-t(use2)  ##nodes for the full tree##
colnames(stats3)=colnames(tuse)
colnames(splits3)=colnames(tuse)
colnames(vars3)=colnames(tuse)

tnodes<-matrix(c(0),nrow=nrow(data2))
for(j in 2:last){
for(i in 1:nrow(data2)){
```

```
if(use[i,j]==0 & tnodes[i]==0)tnodes[i]=use[i,(j-1)]
}
}
for(i in 1:nrow(data2)){
if(use[i,sum(last)]!=0)tnodes[i]=use[i,j]
}
list(nodenumbers=tuse,variables=vars3,splits=splits3,stats=stats3,termnodes=tnodes)
}
```

A.2 TIME-DEPENDENT TSSA

```
library(rpart)
library(survival)

###Time-Dependent Two-Sample Test##

##OUTPUTS KM SURVIVAL DISTRIBUTION ESTIMATES AT EACH EVENT TIME POINT##
##ACCOMODATES LONG FORM OF DATASET WITH TIME-DEPENDENT COVARIATE##
##OUTPUTS THE SET OF FAILURE TIMES ALONG WITH THE KM ESTIMATE FOR EACH TIME POINT##

tdsurv<-function(datatd2,name){

##datatd2: long form dataset, event and time columns labeled "event" and "time"
##name: column name for survival estimates: eg: "surival_est"

if(sum(datatd2$event)!=0){
failtimes<-sort(unique(datatd2$time[datatd2$event==1]))
d<-matrix(c(0),nrow=length(failtimes))
Y<-matrix(c(0),nrow=length(failtimes))
```

```
S1<-matrix(c(0),nrow=length(failtimes))
S<-matrix(c(1),nrow=length(failtimes))
for(f in 1:length(failtimes)){
d[f]=sum(datatd2$event[datatd2$time==failtimes[f]])
Y[f]=length(unique(datatd2$id[datatd2$time>=failtimes[f]]))
S1[f]=1-d[f]/Y[f]
}
S[1]=S1[1]
if(length(failtimes)>1){
for(f in 2:length(failtimes)){
S[f]=S[f-1]*S1[f]
}
}
Sdat<-cbind.data.frame(failtimes,S)
colnames(Sdat)=c("failtimes",paste(name))
return(Sdat)
}
if(sum(datatd2$event)==0){
Sdat=data.frame(matrix(c(1),ncol=2,nrow=1))
colnames(Sdat)=c("failtimes",paste(name))
Sdat
}
}

##OUTPUTS KM CENSORING DISTRIBUTION ESTIMATES AT EACH EVENT TIME POINT##
##ACCOMODATES LONG FORM OF DATASET WITH TIME-DEPENDENT COVARIATE##
##OUTPUTS THE SET OF FAILURE TIMES ALONG WITH THE KM ESTIMATE FOR EACH TIME POINT##

tdcens<-function(datatd2,name){
```

```
##datatd2: long form dataset, event and time columns labeled "event" and "time"
##name: column name for survival estimates: eg: "cens_est"

cens<-matrix(c(0),nrow=length(datatd2$event))
cens[datatd2$event==0 & datatd2$last==1]=1
if(sum(cens)!=0){
failtimes<-sort(unique(datatd2$time[cens==1]))
d<-matrix(c(0),nrow=length(failtimes))
Y<-matrix(c(0),nrow=length(failtimes))
S1<-matrix(c(0),nrow=length(failtimes))
S<-matrix(c(1),nrow=length(failtimes))
for(f in 1:length(failtimes)){
d[f]=sum(cens[datatd2$time==failtimes[f]])
Y[f]=length(unique(datatd2$id[datatd2$time>=failtimes[f]]))
S1[f]=1-d[f]/Y[f]
}
S[1]=S1[1]
if(length(failtimes)>1){
for(f in 2:length(failtimes)){
S[f]=S[f-1]*S1[f]
}
}
Sdat<-cbind.data.frame(failtimes,S)
colnames(Sdat)=c("failtimes",paste(name))
return(Sdat)
}
if(sum(cens)==0){
Sdat=data.frame(matrix(c(1),ncol=2,nrow=1))
colnames(Sdat)=c("failtimes",paste(name))
Sdat
```

```
}
}

##FUNCTION TO GET THE BEST LOG RANK SPLITTING STATISTIC##
##FOR ONE COVARIATE (FIXED OR TIME-DEPENDENT)##
##RETURNS THE SPLITTING VALUE, THE SPLITTING STATISTIC##
##AND THE SELECTED COVARIATE (K)

tdlrsplit<-function(datatd2,k,timecol,dcol,min){

##datatd2: long form of dataset
##k: column of covariate of interest
##timecol: column holding observation times
##dcol: column holding event indicator (1=event at that time, 0=no event at that time)
##min: minimum number of individuals in each node##

bigN<-nrow(datatd2)
chi<-matrix(c(0),nrow=bigN)
same<-matrix(c(0),nrow=bigN)
splitp<-matrix(c(0),nrow=bigN)
datasort<-data.frame(datatd2[order(datatd2[,k]),])
failtimes<-unique(datasort[datasort[,dcol]==1,timecol])

if(sum(datasort[,dcol])>0 & sum(datasort[,dcol])<nrow(datasort)){
for(p in min:(bigN-min)){
if(datasort[p,k]!=datasort[(p+1),k]){
m<-matrix(c(0),nrow=length(failtimes))
x<-matrix(c(0),nrow=length(failtimes))
n<-matrix(c(0),nrow=length(failtimes))
N<-matrix(c(0),nrow=length(failtimes))
```

```
E<-matrix(c(0),nrow=length(failtimes))
V<-matrix(c(0),nrow=length(failtimes))
cut<-datasort[p,k]
for(j in 1:length(failtimes)){
datasort2<-datasort[datasort[,timecol]==failtimes[j],]
##above is the risk set##
split<-matrix(c(0),nrow=nrow(datasort2))
split[datasort2[,k]<=cut]=1
split[datasort2[,k]>cut]=0
m[j]<-sum(split)
##above is the # of individuals in the risk set with values <= the cut-point
x[j]<-sum(split[datasort2[,dcol]==1 & datasort2[,timecol]==failtimes[j]])
##event and value <=cut##
n[j]<-sum(datasort2[datasort2[,timecol]==failtimes[j],dcol]) ##num. of events##
N[j]<-nrow(datasort2)  ##size of the risk set
if(N[j]>1){
E[j]<-m[j]*n[j]/N[j]
V[j]<-(m[j]*n[j]*(N[j]-m[j])*(N[j]-n[j]))/((N[j]-1)*N[j]^2)
}
}
top<-sum(x-E)
bottom2<-sum(V)
if(top!=0 & bottom2!=0)chi[p]<-abs(top/sqrt(sum(V)))
if(top==0 | bottom2==0)chi[p]<-0
}
}
useme<-cbind(chi,datasort[,c(1,k)])
useme2<-useme[order(useme[,2]),]
for(p in min:(bigN-min)){
splitL<-useme2[useme2[,3]<=datasort[p,k],]
```

```
lenL<-splitL$id
lenL2<-length(unique(lenL))
splitR<-useme2[useme2[,3]>datasort[p,k],]
lenR<-splitR$id
lenR2<-length(unique(lenR))
if(lenL2>=min & lenR2>=min & datasort[p,k]!=datasort[p+1,k])
splitp[p]<-(datasort[p,k]+datasort[(p+1),k])/2
}
both<-cbind.data.frame(datasort,chi,splitp)
if(sum(both$splitp)!=0){
buse<-both[both$splitp!=0,]
both3<-buse[buse$chi==max(buse$chi),]
fsplit<-both3$splitp
fsplit<-fsplit[1]
fchi<-both3$chi[1]
return(c(fsplit,fchi,k))
}
if(sum(both$splitp)==0)return(c(0,0,0))
}
if(sum(datasort[,dcol])==0)return(c(0,0,0))
if(sum(datasort[,dcol])==nrow(datasort))return(c(0,0,0))
}

##CALCULATE THE SPLIT STATISTIC BASED ON A##
## PRESELECTED SPLITTING VALUE OF A COVARIATE K##
##USE DURING PRUNING PROCESS TO CALCULATE##
##THE OVERALL PROGNOSTIC ABILITY OF A TREE##

tdlrprune<-function(datatd2,k,timecol,dcol,splitval){
```

```
##datatd2: long form of dataset--typically independent##
##from the dataset used to create the original tree##
##k: column of covariate of interest
##timecol: column holding observation times
##dcol: column holding event indicator (1=event at that time, 0=no event at that time)
##splitval: preselected splitting value##

bigN<-nrow(datatd2)
datasort<-data.frame(datatd2[order(datatd2[,k]),])
failtimes<-unique(datasort[datasort[,dcol]==1,timecol])

if(sum(datasort[,dcol])>0){
##We do not want all individuals without events--this gives no information!!)##
m<-matrix(c(0),nrow=length(failtimes))
x<-matrix(c(0),nrow=length(failtimes))
n<-matrix(c(0),nrow=length(failtimes))
N<-matrix(c(0),nrow=length(failtimes))
E<-matrix(c(0),nrow=length(failtimes))
V<-matrix(c(0),nrow=length(failtimes))
cut<-splitval
for(j in 1:length(failtimes)){
datasort2<-datasort[datasort[,timecol]==failtimes[j],] ##this is the risk set##
split<-matrix(c(0),nrow=nrow(datasort2))
split[datasort2[,k]<=cut]=1
split[datasort2[,k]>cut]=0
m[j]<-sum(split) ##number of individuals in the risk set with values <= the cut-pt
x[j]<-sum(split[datasort2[,dcol]==1
& datasort2[,timecol]==failtimes[j]]) ##event, value<=cutpt##
n[j]<-sum(datasort2[datasort2[,timecol]==failtimes[j],dcol]) ##number of events##
N[j]<-nrow(datasort2)  ##size of the risk set
```

```
if(N[j]>1){
E[j]<-m[j]*n[j]/N[j]
V[j]<-(m[j]*n[j]*(N[j]-m[j])*(N[j]-n[j]))/((N[j]-1)*N[j]^2)
}
}
top<-sum(x-E)
bottom2<-sum(V)
if(top!=0 & bottom2!=0)chi<-abs(top/sqrt(sum(V)))
if(top==0 | bottom2==0)chi<-0
return(chi)
}
if(sum(datasort[,dcol])==0)chi=0
}



##GROW THE ORIGINAL TREE##
##OUTPUT THE SET OF ALPHAS TO BE USED IN BOOTSTRAP PRUNING##

getalphas<-function(datatd,timecol,dcol,tdcovs,minbucket,minsplit,ps=c(1,2)){

##datad: main dataset##
##timecol: column that holds the observation times##
##dcol: column that holds the event indicator##
##(1=event at that time, 0=no event at that time)##
##tdcovs: position of time-dependent covariate##
##minbucket: the minimum # of individuals##
##(or pseudo-subjects) in a terminal node##
##minsplit: the minimum number of individuals##
##(or pseudo-subjects) in a node to allow it to split##
```

```
##ps: pseudo-subject method: 1->make as many pseudo-subjects##
##as necessary, 2<-make only 2 pseudo-subjects##

simvar<-matrix(c(0),nrow=4)
simsplit<-matrix(c(0),nrow=4)
simstat<-matrix(c(0),nrow=4)

i=1
j=2
l=1
nodes<-matrix(c(0),nrow=nrow(datatd),ncol=20)
splits<-matrix(c(0),nrow=nrow(datatd),ncol=20)
vars<-matrix(c(0),nrow=nrow(datatd),ncol=20)
stats<-matrix(c(0),nrow=nrow(datatd),ncol=20)
IDS<-datatd$id

##find best root node split for all data for covariates in vector 'tdcovs'###

tempout<-matrix(c(0),nrow=length(tdcovs),ncol=3)
m=1
for(k in tdcovs){
tempout[m,]<-tdlrsplit(datatd,k,timecol,dcol,minbucket)
m=m+1
}
tempout2<-tempout[max(tempout[,2])==tempout[,2],]
stat1<-as.numeric(tempout2[2])
split1<-as.numeric(tempout2[1])
var1<-as.numeric(tempout2[3])

s<-.01
```

```
##save split info##
nodes[datatd[,var1]<=split1,l]=j
nodes[datatd[,var1]>split1,l]=j+1

if(ps==1)datatd$id[datatd[,var1]>split1]=datatd$id[datatd[,var1]>split1]+s

if(ps==2){
inc=.01
for(z in 1:(nrow(datatd)-1)){
if(IDS[z]==IDS[z+1]){
if(datatd[z,var1]>split1 & datatd[(z+1),var1]<=split1)datatd$id[z+1]=datatd$id[z]+inc
if(datatd[z,var1]<=split1 & datatd[(z+1),var1]>split1)datatd$id[z+1]=datatd$id[z]+inc
if(datatd[z,var1]>split1 & datatd[(z+1),var1]>split1)datatd$id[z+1]=datatd$id[z]
if(datatd[z,var1]<=split1 & datatd[(z+1),var1]<=split1)datatd$id[z+1]=datatd$id[z]
}
}
}

splits[,l]=split1
vars[,l]=var1
stats[,l]=stat1

simvar[i]=var1
simsplit[i]=split1
simstat[i]=stat1

##find best splits for descendant nodes##
i=2
j=4
sum=1
```

```
cont=1
while(cont==1){
if(i>sum){
l=l+1
sum<-sum+2^(l-1)
}
thing<-datatd[nodes[,(l-1)]==i,]
thing2<-length(unique(thing$id))
if(thing2>(minsplit)){
##this makes sure there is enough observations to even try to split a node##
tempout<-matrix(c(0),nrow=length(tdcovs),ncol=3)
m=1
for(k in tdcovs){
tempout[m,]<-tdlrsplit(datatd[nodes[,(l-1)]==i,],k,timecol,dcol,minbucket)
m=m+1
}
biggest<-tempout[max(tempout[,2])==tempout[,2],2]
tempout2<-tempout[max(tempout[,2])==tempout[,2],]
if(length(biggest)>1){
len<-nrow(tempout2)
what<-sample(1:len,1)
tempout2<-tempout2[what,]
}
stat1<-as.numeric(tempout2[2])
split1<-as.numeric(tempout2[1])
var1<-as.numeric(tempout2[3])

if(sum(stat1)!=0){
nodes[datatd[,var1]<=split1 & nodes[,(l-1)]==i,l]=j
nodes[datatd[,var1]>split1 & nodes[,(l-1)]==i,l]=j+1
```

```
if(ps==1)datatd$id[datatd[,var1]>split1 & nodes[,(l-1)]==i]
   =datatd$id[datatd[,var1]>split1 & nodes[,(l-1)]==i]+s
if(ps==2){
inc=.01
dataids<-datatd[nodes[,(l-1)]==i,]
for(z in 1:(nrow(dataids)-1)){
IDS2<-IDS[nodes[,(l-1)]==i]
if(IDS2[z]==IDS2[z+1]){
if(dataids[z,var1]>split1 & dataids[(z+1),var1]<=split1)
dataids$id[z+1]=dataids$id[z]+inc
if(dataids[z,var1]<=split1 & dataids[(z+1),var1]>split1)
dataids$id[z+1]=dataids$id[z]+inc
if(dataids[z,var1]>split1 & dataids[(z+1),var1]>split1)
dataids$id[z+1]=dataids$id[z]
if(dataids[z,var1]<=split1 & dataids[(z+1),var1]<=split1)
dataids$id[z+1]=dataids$id[z]
}
}
datatd[nodes[,(l-1)]==i,]<-dataids
}
splits[nodes[,(l-1)]==i,l]=split1
vars[nodes[,(l-1)]==i,l]=var1
stats[nodes[,(l-1)]==i,l]=stat1
if(i==2 | i==3){
if(i==2)k=2
if(i==3)k=3
simvar[k]=var1
simsplit[k]=split1
simstat[k]=stat1
```

```
}
}
}
i=i+1
j=j+2
kg<-table(nodes[,(l-1)])
kg2<-kg[rownames(kg)!="0"]
kg2[kg2>=(minsplit)]=9999
cont[sum(kg2)<9999]=0
}

last=0
cs<-colSums(nodes)
last<-length(cs[cs!=0])  ##number of layers in tree##
use<-nodes[,1:(last)]
rownames(use)=seq(1:nrow(use))
use2<-unique(use)
splits2<-splits[,1:(last)]
splits3<-t(splits2[c(as.matrix(as.numeric(rownames(use2)))),])
vars2<-vars[,1:(last)]
vars3<-t(vars2[as.matrix(as.numeric(rownames(use2))),])
stats2<-stats[,1:(last)]
stats3<-t(stats2[as.matrix(as.numeric(rownames(use2))),])
tuse<-t(use2)  ##nodes for the full tree##
colnames(stats3)=colnames(tuse)
colnames(splits3)=colnames(tuse)
colnames(vars3)=colnames(tuse)
tnodes<-matrix(c(0),nrow=nrow(datatd))
for(j in 2:last){
for(i in 1:nrow(datatd)){
```

```
if(use[i,j]==0 & tnodes[i]==0)tnodes[i]=use[i,(j-1)]
}
}
for(i in 1:nrow(datatd)){
if(use[i,sum(last)]!=0)tnodes[i]=use[i,j]
}

###now prune the tree##

last=0
cs<-colSums(nodes)
last<-length(cs[cs!=0])  ##number of layers##
use<-nodes[,1:(last)]
rownames(use)=seq(1:nrow(use))
use2<-unique(use)
left<-use2%%2
left[use2==0]=-99  ##left is a matrix to indicate whether node goes to left or right##
str<-t(left)  ##structure of full tree##
alphavals<-matrix(c(0),nrow=length(splits3[splits3!=0]))
minnode<-matrix(c(0),nrow=ncol(splits3))
Gmax<-c(-999)
tuse.w<-rbind(matrix(c(1),ncol=ncol(tuse)),tuse)
stats3.w<-rbind(stats3,matrix(c(0),ncol=ncol(tuse)))
splits3.w<-rbind(splits3,matrix(c(0),ncol=ncol(tuse)))
vars3.w<-rbind(vars3,matrix(c(0),ncol=ncol(tuse)))
str.w<-rbind(str,matrix(c(-99),ncol=ncol(tuse)))
internal<-tuse.w
internal[stats3.w==0]=0
internal2<-t(unique(t(internal)))
stats3.w2<-stats3.w[,colnames(internal2)]
```

```
check<-matrix(c(0),ncol=ncol(internal2),nrow=nrow(internal2))
keepcol<-matrix(c(0),nrow=ncol(internal2))
for(j in 1:ncol(internal2)){
lenval<-matrix(c(-99),nrow=nrow(internal2))
for(i in 1:nrow(internal2)){
value<-internal2[i,j]
if(value!=0)lenval[i]<-length(internal2[i,internal2[i,]==value])
}
if(any(lenval==1))keepcol[j]<-1
}
tuseprune<-internal2[,keepcol==1]
sprune<-cbind.data.frame(stats3.w2[,keepcol==1])
stu=2
G<-matrix(c(0),nrow=length(internal[internal!=0]))
for(a in 1:length(internal[internal!=0])){
if(stu>1){
alpha<-matrix(c(0),nrow=(nrow(sprune)),ncol=ncol(sprune))
mina<-200
##do this if there is more than one path of internal nodes to follow##
if(ncol(sprune)==1){
len<-length(sprune[sprune!=0])  ##number of internal nodes in branch##
if(len>1){
for(i in 1:(len)){ ##i is the node in branch j##
alpha[i]<-sum(sprune[(i:len),])/(length(i:len))
##above: alpha needed to gain prognostic structure for node i in branch j.##
##'stats' is the max tree less any previous pruned branches from other 'a's##
if(alpha[i]<=mina){
##record the minimum alpha and which branch/node it is associated with##
mina=alpha[i]
mini<-i
```

```
minlen<-len
}
}
}
alphavals[a]<-mina
if(mina<200){
minnode[a]<-tuseprune[mini]
prune<-matrix(c(0),nrow=nrow(sprune))
if(tuseprune[mini]==minnode[a])prune[(mini):nrow(prune)]=1
sprune[prune==1]=0
##above:if prune=1, the stats, splits, and covs=zero##
tuseprune[prune==1]=0
stu<-sum(tuseprune)
##now redo the original "tuse" that shows each terminal node##
prune2<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
prune3<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
for(m in 1:ncol(prune2)){
if(tuse.w[mini,m]==minnode[a])prune2[(mini+1):nrow(prune2),m]=1
if(tuse.w[mini,m]==minnode[a])prune3[(mini):nrow(prune2),m]=1
}
stats3.w[prune3==1]=0
##if prune=1, the stats, splits, and covs=zero##
splits3.w[prune3==1]=0
vars3.w[prune3==1]=0
str.w[prune3==1]=-99
tuse.w[prune2==1]=0
}
}
if(ncol(sprune)>1){
for(j in 1:ncol(alpha)){
```

```
##j is number of branches##
len<-length(sprune[sprune[,j]!=0,j])
##above: number of internal nodes in branch##
if(len>1){
for(i in 1:(len)){ ##i is the node in branch j##
alpha[i,j]<-sum(sprune[(i:len),j])/(length(i:len))
##alpha needed to gain prognostic structure for node i in branch j.##
##'stats' is the max tree minus any previous pruned branches##
if(alpha[i,j]<=mina){
##above: record the minimum alpha and which branch/node it is associated with##
mina=alpha[i,j]
mini<-i
minj<-j
minlen<-len
}
}
}
}
alphavals[a]<-mina
if(mina<200){
minnode[a]<-tuseprune[mini,minj]
##above; node in the subtree 'a' to be pruned off##
prune<-matrix(c(0),nrow=(nrow(sprune)),ncol=ncol(sprune))
for(m in 1:ncol(prune)){
if(tuseprune[mini,m]==minnode[a])prune[(mini):nrow(prune),m]=1
}
sprune[prune==1]=0
##above: if prune=1, the stats, splits, and covs=zero##
tuseprune[prune==1]=0
keepcol<-matrix(c(0),nrow=ncol(tuseprune))
```

```
for(j in 1:ncol(tuseprune)){
lenval<-matrix(c(-99),nrow=nrow(tuseprune))
for(i in 1:nrow(tuseprune)){
value<-tuseprune[i,j]
if(value!=0)lenval[i]<-length(tuseprune[i,tuseprune[i,]==value])
}
if(any(lenval==1))keepcol[j]<-1
}
tuseprune<-tuseprune[,keepcol==1]
sprunenew<-sprune[,keepcol==1]
cnames<-colnames(sprune)
cnames2<-cnames[keepcol==1]
sprunenew2<-data.frame(sprunenew)
colnames(sprunenew2)<-cnames2
sprune<-sprunenew2
stu<-sum(tuseprune)

##now redo the original "tuse" that shows each terminal node##
prune2<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
prune3<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
for(m in 1:ncol(prune2)){
if(tuse.w[mini,m]==minnode[a])
prune2[(mini+1):nrow(prune2),m]=1
if(tuse.w[mini,m]==minnode[a])
prune3[(mini):nrow(prune2),m]=1
}
stats3.w[prune3==1]=0
##above: if prune=1, the stats, splits, and covs=zero##
splits3.w[prune3==1]=0
vars3.w[prune3==1]=0
```

```
str.w[prune3==1]=-99
tuse.w[prune2==1]=0
}
}
}
}
return(alphavals[1:a])
}

##GROW TREE AND THEN GET THE SET OF STATISTICS QUANTIFYING
## THE VALUE OF EACH SUBTREES CREATED BY EACH ALPHAS##
##USE TO CREATE EMPIRICAL DISTRIBUTION OF THE BIAS OF
##USING THE SAME SAMPLE VS. A BOOTSTRAP SAMPLE WHEN PRUNING THE TREE##
##OUTPUTS THE SET OF STATISTICS QUANTIFYING THE SUBTREE USING EACH ALPHA##

alphaboots<-function(datatd,dataprune,alphas2,timecol,dcol,
tdcovs,minbucket,minsplit,ps=c(1,2)){

##datatd: dataset (long form) used to grow tree
##dataprune: dataset (long form) used to prune tree
##alphas2: the set of alphas selected by the function "getalphas()"
##timecol: column holding the times of each observation
##dcol: column identifying if there was an event (1) or censoring (0)
##tdcovs: column with time-dependent covariate
##minbucket: minimum number of observations in each node
##minsplit: minimum number of observations to consider splitting a node
##ps: pseudo-subject method, 1=create as many as necessary, 2=create only 2

simvar<-matrix(c(0),nrow=4)
simsplit<-matrix(c(0),nrow=4)
```

```
simstat<-matrix(c(0),nrow=4)
i=1
j=2
l=1
nodes<-matrix(c(0),nrow=nrow(datatd),ncol=20)
splits<-matrix(c(0),nrow=nrow(datatd),ncol=20)
vars<-matrix(c(0),nrow=nrow(datatd),ncol=20)
stats<-matrix(c(0),nrow=nrow(datatd),ncol=20)
IDS<-datatd$id

##find best root node split for all data for covariates in vector 'tdcovs'###

tempout<-matrix(c(0),nrow=length(tdcovs),ncol=3)
m=1
for(k in tdcovs){
tempout[m,]<-tdlrsplit(datatd,k,timecol,dcol,minbucket)
m=m+1
}
tempout2<-tempout[max(tempout[,2])==tempout[,2],]
stat1<-as.numeric(tempout2[2])
split1<-as.numeric(tempout2[1])
var1<-as.numeric(tempout2[3])
s<-.01
##save split info##
nodes[datatd[,var1]<=split1,l]=j
nodes[datatd[,var1]>split1,l]=j+1

if(ps==1)datatd$id[datatd[,var1]>split1]=
datatd$id[datatd[,var1]>split1]+s
if(ps==2){
```

```
inc=.01
for(z in 1:(nrow(datatd)-1)){
if(IDS[z]==IDS[z+1]){
if(datatd[z,var1]>split1 & datatd[(z+1),var1]<=split1)
datatd$id[z+1]=datatd$id[z]+inc
if(datatd[z,var1]<=split1 & datatd[(z+1),var1]>split1)
datatd$id[z+1]=datatd$id[z]+inc
if(datatd[z,var1]>split1 & datatd[(z+1),var1]>split1)
datatd$id[z+1]=datatd$id[z]
if(datatd[z,var1]<=split1 & datatd[(z+1),var1]<=split1)
datatd$id[z+1]=datatd$id[z]
}
}
}

splits[,l]=split1
vars[,l]=var1
stats[,l]=stat1
simvar[i]=var1
simsplit[i]=split1
simstat[i]=stat1

##find best splits for descendant nodes##
i=2
j=4
sum=1
cont=1
while(cont==1){
if(i>sum){
l=l+1
```

```
sum<-sum+2^(l-1)
}
thing<-datatd[nodes[,(l-1)]==i,]
thing2<-length(unique(thing$id))
if(thing2>(minsplit)){
##above: this makes sure there are enough obs to even try to split a node##
tempout<-matrix(c(0),nrow=length(tdcovs),ncol=3)
m=1
for(k in tdcovs){
tempout[m,]<-tdlrsplit(datatd[nodes[,(l-1)]==i,],k,timecol,dcol,minbucket)
m=m+1
}
biggest<-tempout[max(tempout[,2])==tempout[,2],2]
tempout2<-tempout[max(tempout[,2])==tempout[,2],]
if(length(biggest)>1){
len<-nrow(tempout2)
what<-sample(1:len,1)
tempout2<-tempout2[what,]
}
stat1<-as.numeric(tempout2[2])
split1<-as.numeric(tempout2[1])
var1<-as.numeric(tempout2[3])

if(sum(stat1)!=0){
nodes[datatd[,var1]<=split1 & nodes[,(l-1)]==i,l]=j
nodes[datatd[,var1]>split1 & nodes[,(l-1)]==i,l]=j+1
if(ps==1)datatd$id[datatd[,var1]>split1 & nodes[,(l-1)]==i]
  =datatd$id[datatd[,var1]>split1 & nodes[,(l-1)]==i]+s
if(ps==2){
inc=.01
```

```
dataids<-datatd[nodes[,(l-1)]==i,]
for(z in 1:(nrow(dataids)-1)){
IDS2<-IDS[nodes[,(l-1)]==i]
if(IDS2[z]==IDS2[z+1]){
if(dataids[z,var1]>split1 & dataids[(z+1),var1]
   <=split1)dataids$id[z+1]=dataids$id[z]+inc
if(dataids[z,var1]<=split1 & dataids[(z+1),var1]
   >split1)dataids$id[z+1]=dataids$id[z]+inc
if(dataids[z,var1]>split1 & dataids[(z+1),var1]
   >split1)dataids$id[z+1]=dataids$id[z]
if(dataids[z,var1]<=split1 & dataids[(z+1),var1]
   <=split1)dataids$id[z+1]=dataids$id[z]
}
}
datatd[nodes[,(l-1)]==i,]<-dataids
}
splits[nodes[,(l-1)]==i,l]=split1
vars[nodes[,(l-1)]==i,l]=var1
stats[nodes[,(l-1)]==i,l]=stat1
if(i==2 | i==3){
if(i==2)k=2
if(i==3)k=3
simvar[k]=var1
simsplit[k]=split1
simstat[k]=stat1
}
}
}
i=i+1
j=j+2
```

```
kg<-table(nodes[,(l-1)])
kg2<-kg[rownames(kg)!="0"]
kg2[kg2>=(minsplit)]=9999
cont[sum(kg2)<9999]=0
}

last=0
cs<-colSums(nodes)
last<-length(cs[cs!=0])  ##number of layers##
use<-nodes[,1:(last)]
rownames(use)=seq(1:nrow(use))
use2<-unique(use)
splits2<-splits[,1:(last)]
splits3<-t(splits2[c(as.matrix(as.numeric(rownames(use2)))),])
vars2<-vars[,1:(last)]
vars3<-t(vars2[as.matrix(as.numeric(rownames(use2))),])
stats2<-stats[,1:(last)]
stats3<-t(stats2[as.matrix(as.numeric(rownames(use2))),])
tuse<-t(use2)  ##nodes for the full tree##
colnames(stats3)=colnames(tuse)
colnames(splits3)=colnames(tuse)
colnames(vars3)=colnames(tuse)
tnodes<-matrix(c(0),nrow=nrow(datatd))
for(j in 2:last){
for(i in 1:nrow(datatd)){
if(use[i,j]==0 & tnodes[i]==0)tnodes[i]=use[i,(j-1)]
}
}
for(i in 1:nrow(datatd)){
if(use[i,sum(last)]!=0)tnodes[i]=use[i,j]
```

```
}


###now prune the tree##
last=0
cs<-colSums(nodes)
last<-length(cs[cs!=0])  ##number of layers##
use<-nodes[,1:(last)]
rownames(use)=seq(1:nrow(use))
use2<-unique(use)
left<-use2%%2
left[use2==0]=-99  ##left is a matrix to indicate whether node goes to left or right##
str<-t(left)  ##structure of full tree##
alphavals<-matrix(c(0),nrow=length(splits3[splits3!=0]))
minnode<-matrix(c(0),nrow=ncol(splits3))
Gmax<-c(-999)
alphas2<-unique(alphas2)
G<-matrix(c(0),nrow=length(alphas2))
tuse.w<-rbind(matrix(c(1),ncol=ncol(tuse)),tuse)
stats3.w<-rbind(stats3,matrix(c(0),ncol=ncol(tuse)))
splits3.w<-rbind(splits3,matrix(c(0),ncol=ncol(tuse)))
vars3.w<-rbind(vars3,matrix(c(0),ncol=ncol(tuse)))
str.w<-rbind(str,matrix(c(-99),ncol=ncol(tuse)))
internal<-tuse.w
internal[stats3.w==0]=0
internal2<-t(unique(t(internal)))
stats3.w2<-stats3.w[,colnames(internal2)]
check<-matrix(c(0),ncol=ncol(internal2),nrow=nrow(internal2))
keepcol<-matrix(c(0),nrow=ncol(internal2))
for(j in 1:ncol(internal2)){
lenval<-matrix(c(-99),nrow=nrow(internal2))
```

```
for(i in 1:nrow(internal2)){
value<-internal2[i,j]
if(value!=0)lenval[i]<-length(internal2[i,internal2[i,]==value])
}
if(any(lenval==1))keepcol[j]<-1
}
tuseprune<-internal2[,keepcol==1]
sprune<-cbind.data.frame(stats3.w2[,keepcol==1])
cn3<-colnames(stats3.w2)
cn3b<-cn3[keepcol==1]
colnames(sprune)<-cn3b
stu=2
for(a in 1:(length(alphas2)-1)){
##a denotes the number of nested subtrees##
if(stu>1){
alpha<-matrix(c(0),nrow=(nrow(sprune)),ncol=ncol(sprune))
mina<-200
##do this if there is more than one path of itnernal nodes to follow##
if(ncol(sprune)==1){
len<-length(sprune[sprune!=0])  ##number of internal nodes in branch##
if(len>1){
for(i in 1:(len)){ ##i is the node in branch j##
alpha[i]<-sum(sprune[(i:len),])/(length(i:len))
##alpha needed to gain prognostic structure for node i in branch j.
##'stats' is the max tree minus any previous pruned branches##
if(alphas2[a]<=alpha[i] & alphas2[a+1]>alpha[i]){
##above: record the minimum alpha and which branch/node it is associated with##
mina=alpha[i]
mini<-i
minlen<-len
```

```
}
}
}
alphavals[a]<-mina
if(mina<200){
minnode[a]<-tuseprune[mini]
prune<-matrix(c(0),nrow=nrow(sprune))
if(tuseprune[mini]==minnode[a])prune[(mini):nrow(prune)]=1
sprune[prune==1]=0
##if prune=1, the stats, splits, and covs=zero##
tuseprune[prune==1]=0
stu<-sum(tuseprune)
##now redo the original "tuse" that shows each terminal node##
prune2<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
prune3<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
for(m in 1:ncol(prune2)){
if(tuse.w[mini,m]==minnode[a])prune2[(mini+1):nrow(prune2),m]=1
if(tuse.w[mini,m]==minnode[a])prune3[(mini):nrow(prune2),m]=1
}
stats3.w[prune3==1]=0
##if prune=1, the stats, splits, and covs=zero##
splits3.w[prune3==1]=0
vars3.w[prune3==1]=0
str.w[prune3==1]=-99
tuse.w[prune2==1]=0
}
}
if(ncol(sprune)>1){
for(j in 1:ncol(alpha)){ ##j is number of branches##
len<-length(sprune[sprune[,j]!=0,j])  ##number of internal nodes in branch##
```

```
if(len>1){
for(i in 1:(len)){ ##i is the node in branch j##
alpha[i,j]<-sum(sprune[(i:len),j])/(length(i:len))
##alpha needed to gain prognostic structure for node i in branch j.
##'stats' is the max tree minus any previous pruned branches##
if(alphas2[a]<=alpha[i,j] & alphas2[a+1]>alpha[i,j]){
##above: record the minimum alpha and which branch/node it is associated with##
mina=alpha[i,j]
mini<-i
minj<-j
minlen<-len
}
}
}
}
alphavals[a]<-mina
if(mina<200){
minnode[a]<-tuseprune[mini,minj]
##node in the subtree 'a' to be pruned off##
prune<-matrix(c(0),nrow=(nrow(sprune)),ncol=ncol(sprune))
for(m in 1:ncol(prune)){
if(tuseprune[mini,m]==minnode[a])prune[(mini):nrow(prune),m]=1
}
sprune[prune==1]=0
##if prune=1, the stats, splits, and covs=zero##
tuseprune[prune==1]=0
keepcol<-matrix(c(0),nrow=ncol(tuseprune))
for(j in 1:ncol(tuseprune)){
lenval<-matrix(c(-99),nrow=nrow(tuseprune))
for(i in 1:nrow(tuseprune)){
```

```
value<-tuseprune[i,j]
if(value!=0)lenval[i]<-length(tuseprune[i,tuseprune[i,]==value])
}
if(any(lenval==1))keepcol[j]<-1
}
tuseprune<-tuseprune[,keepcol==1]
sprunenew<-sprune[,keepcol==1]
cnames<-colnames(sprune)
cnames2<-cnames[keepcol==1]
sprunenew2<-data.frame(sprunenew)
colnames(sprunenew2)<-cnames2
sprune<-sprunenew2
stu<-sum(tuseprune)
##now redo the original "tuse" that shows each terminal node##
prune2<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
prune3<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
for(m in 1:ncol(prune2)){
if(tuse.w[mini,m]==minnode[a])prune2[(mini+1):nrow(prune2),m]=1
if(tuse.w[mini,m]==minnode[a])prune3[(mini):nrow(prune2),m]=1
}
stats3.w[prune3==1]=0
##if prune=1, the stats, splits, and covs=zero##
splits3.w[prune3==1]=0
vars3.w[prune3==1]=0
str.w[prune3==1]=-99
tuse.w[prune2==1]=0
}
}
}
data2<-dataprune
```

```
nodes2<-matrix(c(0),nrow=nrow(data2),ncol=20)
splits4<-matrix(c(0),nrow=nrow(data2),ncol=20)
vars4<-matrix(c(0),nrow=nrow(data2),ncol=20)
stats4<-matrix(c(0),nrow=nrow(stats3.w),ncol=ncol(stats3.w))
colnames(stats4)<-colnames(stats3.w)
tempsplit<-matrix(c(0),nrow=nrow(data2))
data3<-cbind.data.frame(data2,tempsplit)
##data for pruning plus a space to put node assignments##
if(sum(tuse.w)!=ncol(tuse.w)){
for(b in 1:ncol(tuse.w)){
##b indicates the columns in the pruned subtree##
nodes2<-matrix(c(0),nrow=nrow(dataprune),ncol=20)
s<-str.w[,b]
bigl<-length(s[s!=-99])
##bigl indicates the nodes in the branch##
i=1
j=2
l=1
data3$tempsplit[data3[,vars3.w[l,b]]<=splits3.w[l,b]]=1
newstat<-tdlrprune(data3,vars3.w[l,b],timecol,dcol,splits3.w[l,b])
nodes2[data3[,vars3.w[l,b]]<=splits3.w[l,b],1]=j
nodes2[data3[,vars3.w[l,b]]>splits3.w[l,b],1]=j+1
stats4[l,b]=newstat
i=i+1
if(bigl>1){
for(l in 2:bigl){
if(str.w[(l-1),b]==0)datanew<-data3[nodes2[,(l-1)]==(i),]
if(str.w[(l-1),b]==1)datanew<-data3[nodes2[,(l-1)]==(i+1),]
splitval<-splits3.w[l,b]
splitcov<-vars3.w[l,b]
```

```
datanew$tempsplit=0
datanew$tempsplit[datanew[,splitcov]<=splitval]=1
if(sum(datanew$tempsplit)<nrow(datanew) &&
  (nrow(datanew)-sum(datanew$tempsplit))<nrow(datanew)){
fit2<-tdlrprune(datanew,splitcov,timecol,dcol,splitval)
newstat<-fit2
if(str.w[(l-1),b]==0){
nodes2[data3[,splitcov]<=splitval & nodes2[,(l-1)]==i,l]=i*2
nodes2[data3[,splitcov]>splitval & nodes2[,(l-1)]==i,l]=i*2+1
i=i*2
}
if(str.w[(l-1),b]==1){
nodes2[data3[,splitcov]<=splitval & nodes2[,(l-1)]==i+1,l]=(i+1)*2
nodes2[data3[,splitcov]>splitval & nodes2[,(l-1)]==i+1,l]=(i+1)*2+1
i=(i+1)*2
}
stats4[l,b]=newstat
}
if(sum(datanew$tempsplit)==0 | sum(datanew$tempsplit)==nrow(datanew))break
}
}
}
if(ncol(sprune)==1){
stats4int<-stats4[,colnames(sprune)]
sum2stat<-sum(stats4int)
}
if(ncol(sprune)>1){
stats4int<-stats4[,colnames(tuseprune)]
sumstat<-matrix(c(0),nrow=nrow(data.frame(tuseprune)))
r=1
```

```
for(x in unique(as.vector(tuseprune))){
sumstat[r]<-unique(stats4int[tuseprune==x])
r=r+1
}
sum2stat<-sum(sumstat)
}
aval=2
bigS<-unique(as.vector(tuseprune))
G[a]=sum2stat-aval*(length(bigS)-1)
##for this subtree 'a', calculate the amount of structure
##(split stat) minus the size of branch.
##we want to maximize this number with the new data##
if(G[a]>Gmax){
Gmax<-G[a]
maxa<-a
besttree<-t(unique(t(tuse.w)))
bestsplits<-splits3.w[,colnames(besttree)]
bestcovs<-vars3.w[,colnames(besttree)]
finalnodes<-nodes2
}
}
}
list(data.frame(G),data.frame(besttree),
data.frame(bestsplits),data.frame(bestcovs))
}

##PROVIDES THE FINAL TREE MODEL BASED ON B BOOTSTRAP SAMPLES
##THAT CALCULATE THE BIAS RESULTING FROM USING THE SAME DATA TO
##GROW THE TREE AND ALSO TO PRUNE THE TREE##
```

```
finalmodel<-function(datatd,dataprune,diff,alphas2,timecol,
dcol,tdcovs,minbucket,minsplit,ps=c(1,2)){

##datatd: dataset (long form) used to grow tree
##dataprune: dataset (long form) used to prune tree--
##should be the same dataset as datatd
##diff: the vector of biase that results from
##alphas2: the set of alphas selected by the function "getalphas()"
##timecol: column holding the times of each observation
##dcol: column identifying if there was an event (1) or censoring (0)
##tdcovs: column with time-dependent covariate
##minbucket: minimum number of observations in each node
##minsplit: minimum number of observations to consider splitting a node
##ps: pseudo-subject method, 1=create as many as necessary, 2=create only 2

simvar<-matrix(c(0),nrow=4)
simsplit<-matrix(c(0),nrow=4)
simstat<-matrix(c(0),nrow=4)

i=1
j=2
l=1
nodes<-matrix(c(0),nrow=nrow(datatd),ncol=20)
splits<-matrix(c(0),nrow=nrow(datatd),ncol=20)
vars<-matrix(c(0),nrow=nrow(datatd),ncol=20)
stats<-matrix(c(0),nrow=nrow(datatd),ncol=20)
IDS<-datatd$id

##find best root node split for all data for covariates in vector 'tdcovs'###
tempout<-matrix(c(0),nrow=length(tdcovs),ncol=3)
```

```
m=1
for(k in tdcovs){
 tempout[m,]<-tdlrsplit(datatd,k,timecol,dcol,minbucket)
 m=m+1
}
tempout2<-tempout[max(tempout[,2])==tempout[,2],]
stat1<-as.numeric(tempout2[2])
split1<-as.numeric(tempout2[1])
var1<-as.numeric(tempout2[3])
s<-.01
##save split info##
nodes[datatd[,var1]<=split1,l]=j
nodes[datatd[,var1]>split1,l]=j+1
if(ps==1)datatd$id[datatd[,var1]>split1]=datatd$id[datatd[,var1]>split1]+s
if(ps==2){
inc=.01
for(z in 1:(nrow(datatd)-1)){
if(IDS[z]==IDS[z+1]){
if(datatd[z,var1]>split1 & datatd[(z+1),var1]
  <=split1)datatd$id[z+1]=datatd$id[z]+inc
if(datatd[z,var1]<=split1 & datatd[(z+1),var1]
  >split1)datatd$id[z+1]=datatd$id[z]+inc
if(datatd[z,var1]>split1 & datatd[(z+1),var1]
  >split1)datatd$id[z+1]=datatd$id[z]
if(datatd[z,var1]<=split1 & datatd[(z+1),var1]
  <=split1)datatd$id[z+1]=datatd$id[z]
}
}
}
```

```
splits[,l]=split1
vars[,l]=var1
stats[,l]=stat1
simvar[i]=var1
simsplit[i]=split1
simstat[i]=stat1
##find best splits for descendant nodes##
i=2
j=4
sum=1
cont=1
while(cont==1){
if(i>sum){
l=l+1
sum<-sum+2^(l-1)
}
thing<-datatd[nodes[,(l-1)]==i,]
thing2<-length(unique(thing$id))
if(thing2>(minsplit)){
##this makes sure there is enough observations to even try to split a node##
tempout<-matrix(c(0),nrow=length(tdcovs),ncol=3)
m=1
for(k in tdcovs){
tempout[m,]<-tdlrsplit(datatd[nodes[,(l-1)]==i,],k,timecol,dcol,minbucket)
m=m+1
}
biggest<-tempout[max(tempout[,2])==tempout[,2],2]
tempout2<-tempout[max(tempout[,2])==tempout[,2],]
if(length(biggest)>1){
len<-nrow(tempout2)
```

```
what<-sample(1:len,1)
tempout2<-tempout2[what,]
}
stat1<-as.numeric(tempout2[2])
split1<-as.numeric(tempout2[1])
var1<-as.numeric(tempout2[3])
if(sum(stat1)!=0){
nodes[datatd[,var1]<=split1 & nodes[,(l-1)]==i,l]=j
nodes[datatd[,var1]>split1 & nodes[,(l-1)]==i,l]=j+1
if(ps==1)datatd$id[datatd[,var1]>split1
  & nodes[,(l-1)]==i]=datatd$id[datatd[,var1]>split1 & nodes[,(l-1)]==i]+s
if(ps==2){
inc=.01
dataids<-datatd[nodes[,(l-1)]==i,]
for(z in 1:(nrow(dataids)-1)){
IDS2<-IDS[nodes[,(l-1)]==i]
if(IDS2[z]==IDS2[z+1]){
if(dataids[z,var1]>split1 & dataids[(z+1),var1]
  <=split1)dataids$id[z+1]=dataids$id[z]+inc
if(dataids[z,var1]<=split1 & dataids[(z+1),var1]
  >split1)dataids$id[z+1]=dataids$id[z]+inc
if(dataids[z,var1]>split1 & dataids[(z+1),var1]
  >split1)dataids$id[z+1]=dataids$id[z]
if(dataids[z,var1]<=split1 & dataids[(z+1),var1]
  <=split1)dataids$id[z+1]=dataids$id[z]
}
}
datatd[nodes[,(l-1)]==i,]<-dataids
}
splits[nodes[,(l-1)]==i,l]=split1
```

```
vars[nodes[,(l-1)]==i,l]=var1
stats[nodes[,(l-1)]==i,l]=stat1
if(i==2 | i==3){
if(i==2)k=2
if(i==3)k=3
simvar[k]=var1
simsplit[k]=split1
simstat[k]=stat1
}
}
}
i=i+1
j=j+2
kg<-table(nodes[,(l-1)])
kg2<-kg[rownames(kg)!="0"]
kg2[kg2>=(minsplit)]=9999
cont[sum(kg2)<9999]=0
##if(l>5)break
}
last=0
cs<-colSums(nodes)
last<-length(cs[cs!=0])  ##number of layers##
use<-nodes[,1:(last)]
rownames(use)=seq(1:nrow(use))
use2<-unique(use)
splits2<-splits[,1:(last)]
splits3<-t(splits2[c(as.matrix(as.numeric(rownames(use2)))),])
vars2<-vars[,1:(last)]
vars3<-t(vars2[as.matrix(as.numeric(rownames(use2))),])
stats2<-stats[,1:(last)]
```

```
stats3<-t(stats2[as.matrix(as.numeric(rownames(use2))),])
tuse<-t(use2)  ##nodes for the full tree##
colnames(stats3)=colnames(tuse)
colnames(splits3)=colnames(tuse)
colnames(vars3)=colnames(tuse)
tnodes<-matrix(c(0),nrow=nrow(datatd))
for(j in 2:last){
for(i in 1:nrow(datatd)){
if(use[i,j]==0 & tnodes[i]==0)tnodes[i]=use[i,(j-1)]
}
}
for(i in 1:nrow(datatd)){
if(use[i,sum(last)]!=0)tnodes[i]=use[i,j]
}
###now prune the tree##
last=0
cs<-colSums(nodes)
last<-length(cs[cs!=0])  ##number of layers##
use<-nodes[,1:(last)]
rownames(use)=seq(1:nrow(use))
use2<-unique(use)
left<-use2%%2
left[use2==0]=-99
##left is a matrix to indicate whether node goes to left or right##
str<-t(left)  ##structure of full tree##
alphavals<-matrix(c(0),nrow=length(splits3[splits3!=0]))
minnode<-matrix(c(0),nrow=ncol(splits3))
Gmax<-c(-999)
alphas2<-unique(alphas2)
G<-matrix(c(0),nrow=length(alphas2))
```

```
tuse.w<-rbind(matrix(c(1),ncol=ncol(tuse)),tuse)
stats3.w<-rbind(stats3,matrix(c(0),ncol=ncol(tuse)))
splits3.w<-rbind(splits3,matrix(c(0),ncol=ncol(tuse)))
vars3.w<-rbind(vars3,matrix(c(0),ncol=ncol(tuse)))
str.w<-rbind(str,matrix(c(-99),ncol=ncol(tuse)))
internal<-tuse.w
internal[stats3.w==0]=0
internal2<-t(unique(t(internal)))
stats3.w2<-stats3.w[,colnames(internal2)]
check<-matrix(c(0),ncol=ncol(internal2),nrow=nrow(internal2))
keepcol<-matrix(c(0),nrow=ncol(internal2))
for(j in 1:ncol(internal2)){
lenval<-matrix(c(-99),nrow=nrow(internal2))
for(i in 1:nrow(internal2)){
value<-internal2[i,j]
if(value!=0)lenval[i]<-length(internal2[i,internal2[i,]==value])
}
if(any(lenval==1))keepcol[j]<-1
}
tuseprune<-internal2[,keepcol==1]
sprune<-cbind.data.frame(stats3.w2[,keepcol==1])
cn3<-colnames(stats3.w2)
cn3b<-cn3[keepcol==1]
colnames(sprune)<-cn3b
stu=2
d2<-as.matrix(diff)
for(a in 1:(length(alphas2)-1)){
##a denotes the number of nested subtrees--why did I choose length(minnode)##
if(stu>1){
alpha<-matrix(c(0),nrow=(nrow(sprune)),ncol=ncol(sprune))
```

```
mina<-200
##do this if there is more than one path of itnernal nodes to follow##
if(ncol(sprune)==1){
len<-length(sprune[sprune!=0])
##number of internal nodes in branch##
if(len>1){
for(i in 1:(len)){ ##i is the node in branch j##
alpha[i]<-sum(sprune[(i:len),])/(length(i:len))
##alpha needed to gain prognostic structure for node i in branch j.##
##'stats' is the max tree minus any previous pruned branches##
if(alphas2[a]<=alpha[i] & alphas2[a+1]>alpha[i]){
##minimum alpha and which branch/node it is associated with##
mina=alpha[i]
mini<-i
minlen<-len
}
}
}
alphavals[a]<-mina
if(mina<200){
minnode[a]<-tuseprune[mini]
prune<-matrix(c(0),nrow=nrow(sprune))
if(tuseprune[mini]==minnode[a])prune[(mini):nrow(prune)]=1
sprune[prune==1]=0
##if prune=1, the stats, splits, and covs=zero##
tuseprune[prune==1]=0
stu<-sum(tuseprune)
##now redo the original "tuse" that shows each terminal node##
prune2<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
prune3<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
```

```
for(m in 1:ncol(prune2)){
if(tuse.w[mini,m]==minnode[a])prune2[(mini+1):nrow(prune2),m]=1
if(tuse.w[mini,m]==minnode[a])prune3[(mini):nrow(prune2),m]=1
}
stats3.w[prune3==1]=0
##if prune=1, the stats, splits, and covs=zero##
splits3.w[prune3==1]=0
vars3.w[prune3==1]=0
str.w[prune3==1]=-99
tuse.w[prune2==1]=0
}
}
if(ncol(sprune)>1){
for(j in 1:ncol(alpha)){
##j is number of branches##
len<-length(sprune[sprune[,j]!=0,j])
##number of internal nodes in branch##
if(len>1){
for(i in 1:(len)){ ##i is the node in branch j##
alpha[i,j]<-sum(sprune[(i:len),j])/(length(i:len))
##alpha needed to gain prognostic structure for node i in branch j. ##
##'stats' is the max tree minus any previous pruned branches##
if(alphas2[a]<=alpha[i,j] & alphas2[a+1]>alpha[i,j]){
##record the minimum alpha and which branch/node it is associated with##
mina=alpha[i,j]
mini<-i
minj<-j
minlen<-len
}
}
```

```
}
}
alphavals[a]<-mina
if(mina<200){
minnode[a]<-tuseprune[mini,minj]
##node in the subtree 'a' to be pruned off##
prune<-matrix(c(0),nrow=(nrow(sprune)),ncol=ncol(sprune))
for(m in 1:ncol(prune)){
if(tuseprune[mini,m]==minnode[a])prune[(mini):nrow(prune),m]=1
}
sprune[prune==1]=0
##if prune=1, the stats, splits, and covs=zero##
tuseprune[prune==1]=0
keepcol<-matrix(c(0),nrow=ncol(tuseprune))
for(j in 1:ncol(tuseprune)){
lenval<-matrix(c(-99),nrow=nrow(tuseprune))
for(i in 1:nrow(tuseprune)){
value<-tuseprune[i,j]
if(value!=0)lenval[i]<-length(tuseprune[i,tuseprune[i,]==value])
}
if(any(lenval==1))keepcol[j]<-1
}
tuseprune<-tuseprune[,keepcol==1]
sprunenew<-sprune[,keepcol==1]
cnames<-colnames(sprune)
cnames2<-cnames[keepcol==1]
sprunenew2<-data.frame(sprunenew)
colnames(sprunenew2)<-cnames2
sprune<-sprunenew2
stu<-sum(tuseprune)
```

```
##now redo the original "tuse" that shows each terminal node##
prune2<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
prune3<-matrix(c(0),nrow=(nrow(tuse.w)),ncol=ncol(tuse.w))
for(m in 1:ncol(prune2)){
if(tuse.w[mini,m]==minnode[a])prune2[(mini+1):nrow(prune2),m]=1
##above shows terminal nodes##
if(tuse.w[mini,m]==minnode[a])prune3[(mini):nrow(prune2),m]=1
}
stats3.w[prune3==1]=0
##if prune=1, the stats, splits, and covs=zero##
splits3.w[prune3==1]=0
vars3.w[prune3==1]=0
str.w[prune3==1]=-99
tuse.w[prune2==1]=0
}
}
##do this if there is just one path of internal nodes to follow##
}
data2<-dataprune
nodes2<-matrix(c(0),nrow=nrow(data2),ncol=20)
splits4<-matrix(c(0),nrow=nrow(data2),ncol=20)
vars4<-matrix(c(0),nrow=nrow(data2),ncol=20)
stats4<-matrix(c(0),nrow=nrow(stats3.w),ncol=ncol(stats3.w))
colnames(stats4)<-colnames(stats3.w)
tempsplit<-matrix(c(0),nrow=nrow(data2))
data3<-cbind.data.frame(data2,tempsplit)
##above: data for pruning plus a space to put node assignments##
if(sum(tuse.w)!=ncol(tuse.w)){
for(b in 1:ncol(tuse.w)){
##b indicates the columns in the pruned subtree##
```

```
nodes2<-matrix(c(0),nrow=nrow(dataprune),ncol=20)
s<-str.w[,b]
bigl<-length(s[s!=-99]) ##bigl indicates the nodes in the branch##
i=1
j=2
l=1
data3$tempsplit[data3[,vars3.w[l,b]]<=splits3.w[l,b]]=1
newstat<-tdlrprune(data3,vars3.w[l,b],timecol,dcol,splits3.w[l,b])
##get value of split statistic for previously-selected split for a node##
nodes2[data3[,vars3.w[l,b]]<=splits3.w[l,b],1]=j
nodes2[data3[,vars3.w[l,b]]>splits3.w[l,b],1]=j+1
stats4[l,b]=newstat
i=i+1
if(bigl>1){
for(l in 2:bigl){
if(str.w[(l-1),b]==0)datanew<-data3[nodes2[,(l-1)]==(i),]
if(str.w[(l-1),b]==1)datanew<-data3[nodes2[,(l-1)]==(i+1),]
splitval<-splits3.w[l,b]
splitcov<-vars3.w[l,b]
datanew$tempsplit=0
datanew$tempsplit[datanew[,splitcov]<=splitval]=1
if(sum(datanew$tempsplit)<nrow(datanew)
  && (nrow(datanew)-sum(datanew$tempsplit))<nrow(datanew)){
fit2<-tdlrprune(datanew,splitcov,timecol,dcol,splitval)
newstat<-fit2
if(str.w[(l-1),b]==0){
nodes2[data3[,splitcov]<=splitval & nodes2[,(l-1)]==i,l]=i*2
nodes2[data3[,splitcov]>splitval & nodes2[,(l-1)]==i,l]=i*2+1
i=i*2
}
```

```
if(str.w[(l-1),b]==1){
nodes2[data3[,splitcov]<=splitval & nodes2[,(l-1)]==i+1,l]=(i+1)*2
nodes2[data3[,splitcov]>splitval & nodes2[,(l-1)]==i+1,l]=(i+1)*2+1
i=(i+1)*2
}
stats4[l,b]=newstat
}
if(sum(datanew$tempsplit)==0 | sum(datanew$tempsplit)==nrow(datanew))break
}
}
}
if(ncol(sprune)==1){
stats4int<-stats4[,colnames(sprune)]
sum2stat<-sum(stats4int)
}
if(ncol(sprune)>1){
stats4int<-stats4[,colnames(tuseprune)]
sumstat<-matrix(c(0),nrow=nrow(data.frame(tuseprune)))
r=1
for(x in unique(as.vector(tuseprune))){
sumstat[r]<-unique(stats4int[tuseprune==x])
r=r+1
}
sum2stat<-sum(sumstat)
}
aval=2
bigS<-unique(as.vector(tuseprune))
G[a]=sum2stat-aval*(length(bigS)-1)+d2[a]
##calculate the amount of structure (split stat)##
## minus the size of branch. Maximize this number with the new data##
```

```
if(G[a]>Gmax){
Gmax<-G[a]
maxa<-a
besttree<-t(unique(t(tuse.w)))
bestsplits<-splits3.w[,colnames(besttree)]
bestcovs<-vars3.w[,colnames(besttree)]
finalnodes<-nodes2
structure<-tuse.w
}
}
}
list(data.frame(G),data.frame(besttree),
data.frame(bestsplits),data.frame(bestcovs),data.frame(structure))
}
```

BIBLIOGRAPHY

[1] Reynolds CFR, Frank E, Perel JM, Imber SD, Cornes C, Miller MD, Mazumdar S, Houck PR, Dew MA, Stack JA, Pollock BJ, and Kupfer DF. Nortiptyline and interpersonal psychotherapy as maintenance therapies for recurrent major depression. *Journal of the Americal Medical Association*, 281(1):39–45, 1999.

[2] Feelders A. Handling missing data in trees: surrogate splits or statistical imputation? In *Principles of Data Mining and Knowledge Discovery*, pages 329–334, 1999.

[3] Little RJA and Rubin DB. *Statistical Analysis with Missing Data*. Wiley Series in Probability and Statistics, Wiley Interscience, New Jersey, 2002.

[4] Therneau TM and Atkinson EJ. *An introduction to Recursive Partitioning Using RPART*.

[5] Conversano C and Siciliano R. Tree-based classifiers for conditional incremental missing data imputation. Technical report, 2003.

[6] Fisher LD and Lin DY. Time-dependent covariates in the cox proportional hazards regression model. *Annual Review of Public Health*, 20, 1999.

[7] Gail MH. Evaluating serial cancer marker studies in patients at risk of recurrent disease. *Biometrics*, 37, 1981.

[8] Graf EA, Schmoor C, Sauerbrei W, and Schumacher M. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine*, 18:2529–2545, 1999.

[9] Bacchetti P and Segal M. Survival trees with time-dependent covariates: Application to estimating chanes in the incubation period of aids. *Lifetime Data Analysis*, 1:35–47, 1995.

[10] Huang X, Chen S, and Soong S. Piecewise exponential survival tree with time-dependent covariates. *Biometrics*, 54:1420–1433, 1998.

[11] Klein J and Moeschberger M. *Survival Analysis: Statistics for biology and health, 2nd ed.* Spring Science+Business Media Inc, New York, 2003.

[12] Kaplan EL and Meier P. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481, 1958.

[13] Cox DR. Regression models and life tables (with discussion). *Journal of the Royal statistical Society, Series B*, 34, 1972.

[14] Cox DR. Partial likelihood. *Biometrika*, 62, 1975.

[15] Morgan JN and Songuist JA. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, June:415–434, 1963.

[16] Ciampi A, Bush RS, Gospodarowics M, and Till JE. An approach to classifying prognostic factors related to survival experience for non-hodgkin's lymphoma patients: Based on a series of 982 patients. *Cancer*, 47, 1981.

[17] Marubini E, Morabito A, and Valsecchi MG. Prognostic factors and risk groups: Some results given by using an algorithm suitable for censored survival data. *Statistics in Medicine*, 2:295–303, 1983.

[18] Breiman L, Freidman JH, Olshen RA, and Stone CJ. *Classification and Regression Trees*. Wadsworth, California, 1984.

[19] Gordon L and Olshen RA. Tree-structured survival analysis. *Cancer Treatment Reports*, 69:1065–1069, 1985.

[20] Davis R and Andersen J. Exponential survival trees. *Statistics in Medicine*, 8:947–961, 1989.

[21] Leblanc M and Crowley J. Relative risk trees for censored survival data. *Biometrics*, 48:411–425, 1992.

[22] Ahn H and Loh W. Tree-structured proportional hazards. *Biometrics*, 50(2):471–485, 1994.

[23] Segal MR. Regression trees for censored data. *Biometrics*, 44:35–47, 1988.

[24] Tarone R and Ware J. On distribution-free tests for equality of survival distributions. *Biometrika*, 64(1):156–160, 1977.

[25] Harrington D and Fleming T. A class of rank test procedures for censored survival data. *Biometrika*, 69(3):553–566, 1982.

[26] LeBlanc M and Crowley J. Survival trees by goodness of split. *Journal of the American Statistical Association*, 88(422):457–467, 1993.

[27] Ciampi A, Hogg S, McKinney S, and Thiffault J. Recpam: A computer program for recursive partition and amalgamation for censored survival data and other situations fre-

quently occurring in biostatistics. i. methods and program features. *Computer Methods and Programs in Biomedicine*, 26:239–256, 1988.

[28] Ciampi A, Abdissa N, and Lou Z. Tree-structured prediction for censored survival data and the cox model. *Journal of Clinical Epidemiology*, 48(5), 1995.

[29] Terry M Therneau and Beth Atkinson. R port by Brian Ripley. *rpart: Recursive Partitioning*, 2008. R package version 3.1-41.

[30] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[31] Mantel N. Evaluation of survival data and two new rank order statistics arising in its consideration. *Cancer Chemotherapy Rep.*, 50:163–170, 1966.

[32] Gehan E. A generalized wilcoxon test for comparing arbitrarily singly-censored samples. *Biometrika*, 52(1/2):203–223, 1965.

[33] Breslow N. A generalized kruskal-wallis test for comparing k samples subject to unequal patterns of censorship. *Biometrika*, 57:579–594, 1970.

[34] Segal MR. Features of tree-structured survival analysis. *Epidemiology*, 8(4):344–346, 1997.

[35] Crawford SL. Extensions to the cart algorithm. *International Journal of Man-Machine Studies*, 31, 1989.

[36] Danneger F. Tree stability diagnostics and some remedies. *Statistics in Medicine*, 19:475–491, 2000.

[37] Hothorn T, Lausen B, Banner A, and Radespiel-Troger M. Bagging survival trees. *Statistics in Medicine*, 23:77–91, 2004.

[38] Schoop R, Graf E, and Schumacher M. Quantifying the predictive performance of prognostic models for censored survival data with time-dependent covariates. *Biometrics*, 64, 2008.

[39] Buyssee DJ, Reynolds CF, Monk TH, Berman SR, and Kupfer DJ. The pittsburgh sleep quality index: a new instrument for psychiatric practice and research. *Psychiatry Reserach*, 28(2):193–213, 1989.

[40] Laird NM and Ware JH. Random-effects models for longitudinal data. *Biometrics*, 38(4), 1982.

[41] Brown H and Prescott R. *Applied Mixed Models in Medicine*. John Wiley and Sons LTD, Chichester, West Sussex, England, 1999.

[42] Fitzmaurice GM, Laird NM, and Ware JH. *Applied Longitudinal Analysis*. John Wiley and Sons, Inc, Hoboken, NJ, 2004.

[43] Henderson CR. *Applications of Linear Models in Animal Breeding*. University of Guelph, Guelph, Ontario, 1984.

[44] Jeong J-H, Jung S, and Costantino JP. Nonparametric inference on median residual life function. *Biometrics*, 64, 2008.

[45] Littell RC, Milliken GA, Stroup WW, and WOlfinger RD. *SAS System for mixed models*. SAS Institute, Cary, NC, 1986.